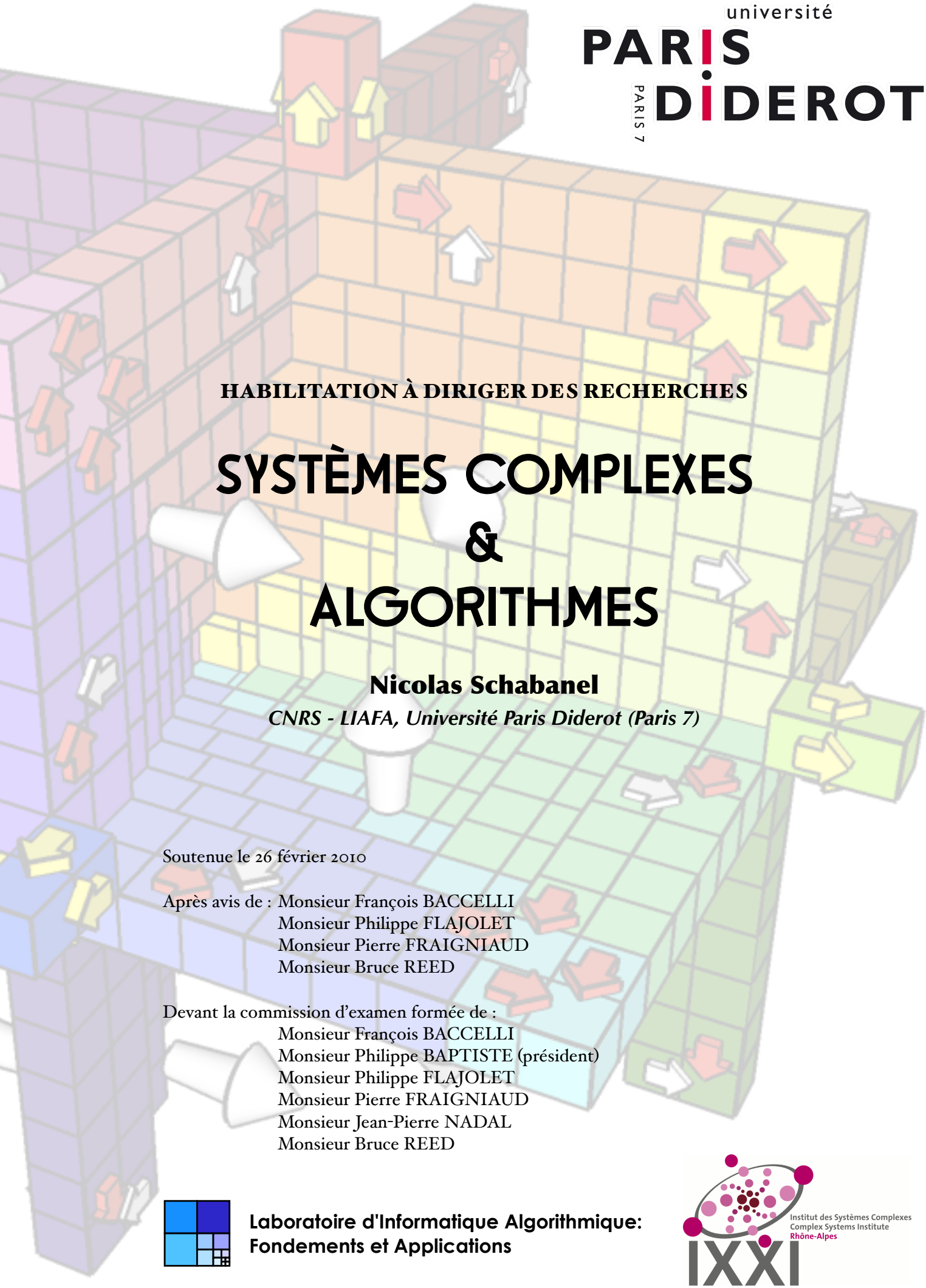


université PARIS DIDEROT PARIS 7

HABILITATION À DIRIGER DES RECHERCHES

SYSTÈMES COMPLEXES & ALGORITHMES

Nicolas Schabanel

CNRS - LIAFA, Université Paris Diderot (Paris 7)

Soutenue le 26 février 2010

Après avis de : Monsieur François BACCELLI
Monsieur Philippe FLAJOLET
Monsieur Pierre FRAIGNIAUD
Monsieur Bruce REED

Devant la commission d'examen formée de :
Monsieur François BACCELLI
Monsieur Philippe BAPTISTE (président)
Monsieur Philippe FLAJOLET
Monsieur Pierre FRAIGNIAUD
Monsieur Jean-Pierre NADAL
Monsieur Bruce REED

Laboratoire d'Informatique Algorithmique: Fondements et Applications

Institut des Systèmes Complexes Complex Systems Institute Rhône-Alpes
IXXI

À Emmanuelle, Damien et Julien

Table des matières

Table des Algorithmes

« J'ai tellement besoin de temps pour ne rien faire,
qu'il ne m'en reste plus assez pour travailler. »

Pierre Reverdy, poète surréaliste (1889-1960)

« Après avoir cherché sans trouver,
il arrive qu'on trouve sans chercher »

Jerome K. Jerome, écrivain et humoriste anglais (1859-1927)

Je suis régulièrement stupéfait par la quantité
de temps que je dois passer à ne rien faire pour finalement arriver
à faire quelque chose, et je ne parle même pas du temps que je
passe à me tromper. Si parfois j'atteins
un rendement de 10% par jour, j'en suis le premier
surpris. J'y ai recherché bien des
remèdes mais sans réel succès : il apparaît que l'inaction
fait partie intégrante de l'action, du moins
pour le domaine de la recherche. Je n'ai pas de mot assez dur
pour exprimer mon agacement envers ceux qui,
les poumons gonflés par l'air ambiant, entendent
faire croire à qui veut
les entendre qu'il ne faut conserver dans ce
métier que ceux dont l'efficacité
est certifiée. La conséquence immédiate de ce harcèlement
est que nous passons une part bien trop importante
de notre temps à écrire et à comparer nos CV, à s'interroger sur
les incroyables performances de tel ou tel,
comme des gamins de dix ans dans une cour
de récréation. On nous gâche ainsi
notre "temps à ne rien faire" si précieux.
Pour moi, le processus de création
reste un mystère où l'inaction joue un rôle moteur. En bref, dans notre métier, mais aussi dans
plein d'autres secteurs, je conclurais à la manière de Pierre Dac que
« ne rien faire, c'est certainement déjà faire quelque chose :
essayez de ne rien faire, ça n'a rien d'évident ! »

Merci

Je tiens tout d'abord à remercier François **Baccelli**, Philippe **Flajolet**, Pierre **Fraigniaud** et Bruce **Reed** d'avoir accepté de relire ce manuscrit ; vos remarques m'ont été très précieuses. Je remercie également Jean-Pierre **Nadal** d'avoir accepté de participer à mon jury et de se plonger dans l'univers informatique et d'y avoir apporté l'éclairage du physicien, en particulier sur des extensions possibles au quatrième chapitre. Je remercie enfin Philippe **Baptiste** d'avoir accepté de présider mon jury.

Emmanuelle, **Damien** et **Julien**, je vous remercie de m'avoir accordé votre confiance et de m'avoir supporté pendant toutes ces années. J'ai tout particulièrement apprécié travailler avec vous et j'ai beaucoup appris sur de nombreux sujets à vos côtés. Je ne peux qu'espérer avoir la chance d'encadrer d'autres thésards aussi géniaux que vous.

De chaleureux mercis à Jacques **Mazoyer** et Marianne **Delorme** pour ces discussions toujours passionnantes, vos nombreux conseils avisés, ces luttes au quotidien et votre joie communicative de la recherche "à l'ancienne" (que tout le monde regrette en secret — *mais, chut ! faut pas le dire*).

Autant de mercis à l'ensemble de l'équipe **MC2** : Éric **Rémila**, Éric **Thierry**, Éric **Boix**, Natacha **Portier**, Pascal **Koiran**, ainsi que tous les thésards qui se sont succédés dont tout particulièrement Nazim **Fatès**, dont la compagnie a grandement contribué à créer une bonne ambiance durant notre séjour dans le carrelage du LR5.

Grands mercis à Michel **Morvan** pour son dynamisme stimulant et exemplaire, les portes qu'il nous a ouvertes, et notre aventure enrichissante de l'**institut des systèmes complexes** et des divers conférences que nous avons organisées ensemble.

Un grand merci à Corinne **Iafrate**, Sylvie **Boyer**, Serge **Torres**, Dominique **Ponsard**, Sarah **Donnelly**, Rebecca **Wright**, Maria Ines **Rivera**, Beatriz **Balmaceda**, Edwige **Royboz** et Noëlle **Delgado** pour votre assistance toujours souriante.

Un grand merci au **département de mathématiques et informatique** (DMI) de l'ÉNS Lyon où j'ai toujours pris plaisir à enseigner dans les meilleures conditions.

I want to thank Fred **Roberts** very much for his kindness during our north-american "misadventures". An uncountable load of thanks to Dr Lawrence **Ettinger and his wife** together with Cathy **Lepeniotis** and Eileen **Cirri** for all of what you've done for Juliette. I was, thanks to all of you, able to get the best from my postdoc at Dimacs in spite of the circumstances. Specials thanks to Yuval and Shlomit **Ishai** and Ashwin **Nayak**.

Miles de gracias a Ivan **Rapaport** (y su mama), Marcos **Kiwi**, Rafael **Correa**, José **Correa**, Julio **Aracena**, Anahí **Gajardo**, Eric **Goles**, Karol **Suchan**, Martín **Matamala**,

Eduardo **Moreno**, Luz **Lindegaard**, Ana **Trujillo**, **Margarita**, **Betty**, **Sophie**, **Albert** y sus hijos.

De nombreux mercis à toutes les personnes qui m'ont aidé/éclairé/décoincé sur des points techniques ou autres à un moment ou un autre, parmi eux : George **Giakkoupis**, Ioan **Todinca**, Michel **Habib**, Cris **Moore**, Mark **Newman**, Emmanuel **Jeandel**, Guillaume **Theyssier**, Nicolas **Hanusse**, Christoph **Dürr**, Laurent **Viennot**, et tout ceux que j'ai oubliés ici.

Merci aux différents laboratoires qui m'ont accueilli et où j'ai toujours pris plaisir à travailler : le **LIP** (Lyon), **LRI** (Orsay), **DIMACS** (Rutgers), l'**IXXI** (Lyon), le **DIM/CMM** (Santiago de Chile), l'**ISCV** (Valparaíso), et le **LIAFA** (Paris). Merci également au **CNRS** pour la très grande liberté qui offre à ses chercheurs, leurs permettant ainsi d'exprimer leur créativité sans (trop d') entraves.

Un grand merci à **mes parents, Monique et François**, d'avoir réussi la performance de relire l'ensemble de ce document pour y traquer fautes d'orthographe et copier-coller malheureux. Un grand merci à **Michel** et **Marie-France** pour le soutien logistique hors pair le jour de la soutenance.

Un immense merci à **Anne, Juliette, Philémon et Hippolyte** d'avoir supporté de manière héroïque et sans trop broncher un papa très occupé pendant cette longue année et demi.

Je remercie enfin le dieu **Deadline** de m'avoir "encouragé" à terminer la rédaction de ce manuscrit dans les "plus brefs délais"...

0 Introduction

« Computer Science is no more about computers than astronomy is about telescopes. »
Edsger W. Dijkstra (1930-2002)

1 Le terreau des systèmes complexes

Les immenses progrès scientifiques jusqu'au milieu du XIXème siècle laissaient entrevoir la possibilité que la science puisse un jour tout expliquer, tout prévoir, tout démontrer, une fois que l'on aura découvert les quelques rouages essentiels (lois, axiomes, etc.) nécessairement simples et en petits nombres qui gouvernent la nature (physique, vivante, ou mathématique, etc.).

1.1 Comportements chaotiques

Ce projet fut bien vite mis à mal entre 1890 et 1898 par les découvertes de Poincaré et d'Hadamard. Poincaré démontra en 1890 [239] [1,2] qu'il existe pour le problème à trois corps en interactions gravitationnelles, des configurations initiales dont l'avenir est totalement imprévisible si elles ne sont pas connues avec une précision infinie : deux trajectoires arbitrairement proches s'éloignent irrémédiablement dans des proportions considérables et imprévisibles (non-linéaires avec le temps). La situation se détériore encore avec le résultat d'Hadamard [156] à propos du billard sur les surfaces à courbures négatives qui démontre que, de presque toute paire de points de départ arbitrairement proches, mais distincts, les trajectoires s'écartent à un rythme exponentiel, rendant toute prédiction à moyen terme impossible si le point de départ est connu avec une précision limitée. Dans les deux cas, nous connaissons pourtant des expressions algébriques des trajectoires (v. [298] et [312] pour les problèmes à trois et n corps, respectivement), mais ces expressions restent totalement inopérantes en pratique (p. ex. pour le problème à n corps, le développement en série entière est obtenu par composition trop près d'un pôle et la moindre erreur s'amplifie).[3] On voit donc

[1] Les nombreuses notes de ce chapitre ont été regroupées aux pages 21 et suivantes.

bien que même une formulation algébrique ne permet pas nécessairement d'en savoir plus. On peut résumer ces résultats de la façon suivante : même pour un système apparemment simple, le connaître avec une précision arbitraire ne suffit pas pour faire des prédictions satisfaisantes sur son évolution.

Poincaré a tout de suite présenti les implications de ces découvertes : *« une cause très petite, qui nous échappe, détermine un effet considérable que nous ne pouvons pas ne pas voir, et alors nous disons que cet effet est dû au hasard »*. Cela signifie que même dans un système totalement déterministe, le hasard peut surgir naturellement, du fait de notre connaissance nécessairement approximative de sa configuration. Cette constatation est à mon avis une justification fort intéressante de l'usage et du succès des probabilités pour modéliser notre monde, comme le souligne Ruelle dans son livre remarquable [276].[4]

1.2 Incomplétude en mathématique

La situation se dégrade encore entre 1931 et 1936 avec la naissance de la *science du calcul* que l'on appellera plus tard l'*informatique*.[5] En 1931, Gödel découvre que l'on peut coder tout texte sous forme d'entiers (une idée bien banale maintenant dans notre monde "tout-numérique")[6] et en déduit que tout système d'axiomes contenant les axiomes définissant les entiers (notre mathématique par exemple) contient nécessairement des théorèmes (énoncés vrais) dont on ne peut pas écrire de preuve, on dit qu'ils sont indécidables [137]. Ce résultat répondit par la négative au second problème de Hilbert sur les fondements de la mathématique [163].

Il n'y a aucun remède : ajoutez un énoncé indécidé comme axiome et vous obtenez de nouveaux énoncés indécidables. De plus, cela implique qu'il est indécidable de déterminer la *cohérence* de la mathématique (c.-à-d., de prouver que l'on ne pourra jamais faire de démonstration correcte de résultats faux). Sinon, il suffirait d'ajouter un énoncé indécidé comme axiome et de tester si le nouveau système est toujours cohérent, pour déterminer si cet énoncé est vrai ou faux.

La mathématique est donc incomplète (on ne peut pas écrire de preuve de certaines propositions vraies) et potentiellement incohérente (il est possible que l'on découvre un jour que notre mathématique puisse conduire à des contradictions).[7]

1.3 Indécidabilité

Le résultat de Gödel est d'autant plus grave que de nombreuses propriétés élémentaires et souhaitables sur le calcul sont indécidables. Par exemple, déterminer si un programme finira par s'arrêter ou non : tout le monde souhaite que son programme de calcul numérique s'arrête et renvoie un résultat, cependant personne ne souhaite que le système d'exploitation d'une centrale nucléaire déclare forfait ; il est malheureusement *impossible* de répondre à ces questions en général.[8]

Ces derniers résultats sont dus à Turing qui introduisit dans [306] un modèle du calcul visuel, simple à comprendre et à utiliser, unifiant les différents modèles de calcul proposés (p. ex., le λ-calcul et les fonctions μ-récursives introduits par Church et Kleene ; v. p. ex. l'article [135] qui retrace les premiers pas vers la définition du calcul). Son modèle, appelé désormais les *machines de Turing*, consiste en une tête de lecture/écrire ayant un nombre fini d'états possibles, et qui se déplace le long d'un ruban arbitrairement grand[9] : à chaque pas de calcul, elle lit la lettre écrite sur le ruban en face de la tête, et en fonction de cette lettre et de son état, écrit

une autre lettre à cet endroit du ruban, change d'état, puis effectue un déplacement (reste sur place, se déplace d'un cran vers la gauche, ou vers la droite), ou bien s'arrête définitivement.[10]

Bien qu'inefficace tel quel en pratique, ce modèle fut un premier pas décisif vers la réalisation des ordinateurs. La *thèse de Church-Turing* affirme qu'il permet d'effectuer absolument tous les calculs *effectifs* possibles, et en particulier d'écrire toutes les preuves correctes possibles en mathématique.[11]

Un des résultats essentiels de Turing est l'existence de machines de Turing *universelles*, c.-à-d., de machines qui peuvent lire sur leur ruban, le code de toute autre machine et d'une entrée, et qui simulent tous les pas de calcul de cette machine sur cette entrée. On peut tirer deux conséquences spectaculaires de ce théorème.

1.4 L'universalité est une propriété banale

Tout d'abord, le fait que l'universalité, c.-à-d., la capacité de faire tout ce qui est possible, est une propriété *élémentaire*, *d'une complexité extrêmement modeste* et donc *inévitable* dès que l'on conçoit un modèle un tant soit peu évolué. Les résultats récents de Theyssier [302] démontrent même que cette propriété est *banale* puisqu'il suffit de choisir un automate cellulaire captif[12] au hasard pour obtenir presque sûrement un automate universel.[13] Pire, le théorème de Rice [260], affirme que *toute* propriété *non-triviale* (terminaison du processus, prédiction de l'apparition d'un certain motif, etc.) est indécidable pour un modèle ayant la capacité universelle.

Cette frontière entre le décidable et l'indécidable mobilise une part importante de la communauté informatique préoccupée de la preuve des programmes[14] dont l'art consiste à vérifier que le programme fait bien ce qu'il doit faire. Pour accomplir cette tâche, il faut absolument rester dans le domaine du décidable, et cela nécessite donc de développer simultanément un langage de programmation original en le limitant et un système de preuves associé en le poussant à ses limites, de façon à assurer que l'on ne sorte pas du domaine décidable pour les propriétés voulues tout en gardant l'expressivité souhaitée.

Plus généralement, on se rend donc bien compte de la difficulté de construire des modèles prédictifs de la nature : pas assez puissants, ils ne modéliserons pas correctement les phénomènes ; trop puissants, on ne pourra rien prédire assurément. On peut citer deux exemples de modèles apparemment simples qui se sont révélés être universels a posteriori : le célèbre jeu de la vie introduit par Conway et popularisé par Gardner en 1970 [132, 43, 91, 259] et la règle **110** de Wolfram, peut-être moins charismatique mais sans aucun doute la plus spectaculaire du fait de son apparente simplicité (deux états, deux voisins, et donc seulement huit transitions différentes possibles !) [72, 261].

1.5 Equivalence entre programmes et données

La deuxième conséquence essentielle, selon moi, de l'existence de machines de Turing universelles est l'équivalence entre programmes et données. Ce sont tous les deux des "bouts de codes", une suite de `0` et de `1` en interactions. Cette idée est trompeusement évidente et ne fut d'ailleurs mise en œuvre que très récemment dans les ordinateurs et pas uniquement à cause de difficultés techniques. Pendant longtemps le programme était chargé à part, sur des cartes, et les données ensuite seulement.

Ce concept est loin d'être aussi trivial qu'il y paraît et les virus informatiques en sont une conséquence intéressante. Une astuce classique pour pénétrer un système est de lui trans-

mettre plus de données qu'il n'en demande, de sorte que si aucune vérification n'est faite sur la longueur de la transmission, vos données vont s'écrire par dessus son propre code et cela va vous permettre de lui faire exécuter le code que vous désirez. Ce concept est aussi à l'œuvre à très grande(s) échelle(s) au niveau du "code génétique" où tout est mêlé : la "mémoire de masse" (les brins d'ADN), le "code" (les brins d'ARN, les positions relatives des parties codantes, les lois chimiques, les molécules décodeuses, ribosomes, etc.), les "données" (les protéines, leurs concentrations, les acides aminés, les conditions "extérieures" au noyau, à la cellule, à l'organe, au corps humain, à son cadre familial, à son cadre social, etc.).[15] On peut d'ailleurs faire un parallèle intéressant entre les fonctionnements des virus biologiques et informatiques : ils détournent à leur propre compte les mécanismes du système parasité par injection de parties codantes. Aucun scientifique ne peut prétendre actuellement qu'obtenir la liste des caractères codés dans l'ADN permettra un jour de décrire ce qui s'y passe précisément ; à l'évidence, cet emmêlement inextricable a sans nul doute la puissance Turing (à un détail près toutefois : les ressources sont finies et clairement délimitées).

1.6 Décidable signifie-t-il prévisible ?

Qu'en est-il des questions décidables ? Une question naturelle est de chercher à prédire dans quel état sera un système au bout d'un certain temps. Les résultats de Poincaré et Hadamard, nous affirment que c'est souvent impossible si on ne connaît pas parfaitement la configuration initiale. Mais *connaître parfaitement la configuration initiale est-il suffisant ?*

Déterminer le temps qu'il fera demain est clairement une question décidable : pour savoir quel temps il fera demain, il suffit d'attendre demain (que le calcul de la nature se fasse). Cependant, ce que l'on souhaiterait c'est *prédire* et par là même espérer comprendre. Dans certain cas, par exemple, il est possible d'aller beaucoup plus vite que la nature, p. ex., on sait calculer la n-ème décimale de π sans avoir à calculer les précédentes [23].

En revanche, un nouveau coup dur fut porté par la démonstration que tout automate pouvant simuler des circuits booléens est P-complet et donc, dans l'état actuel des connaissances, "inaccélérable". Si on trouve un jour un moyen de calculer l'état d'une cellule de cet automate sans passer par le calcul des étapes intermédiaires, on saura le faire pour tous ceux calculables en temps polynomial et le consensus actuel considère que c'est sans doute impossible. Or, le jeu de la vie et bien d'autres modèles "simples" et utiles sont dans ce cas [91, 43, 151, 319].

1.7 À la recherche d'une définition de la complexité

Tous ces résultats posent la question du décalage entre la simplicité de la description d'un système et la complexité potentiellement monstrueuse de ses comportements. Bien sûr, ces résultats négatifs portent en général sur des configurations initiales particulières. Mais bien malin celui qui pourra décrire comment distinguer les configurations initiales problématiques de celles qui ont un comportement agréable. Plusieurs résultats théoriques tentent de comprendre ce décalage.

En 1948, Shannon [288] proposa sa célèbre théorie de l'information permettant de déterminer la quantité d'information contenue dans une suite de caractères. Ce premier pas a été fondamental tant sur le plan pratique (déterminer comment coder efficacement nos voix et nos données pour les transmettre sur des canaux nécessairement imparfaits) que théorique (déterminer quelles sont les parties codantes d'une chaîne d'ADN ou bien au contraire, comment dissimuler l'information de façon à ce qu'elle passe pour du bruit en cryptographie).

En 1965 et 1966, Solomonoff [295], Kolmogorov [191] et Chaitin [61] proposèrent indépendamment une théorie algorithmique de la complexité d'une suite de caractères : l'idée apparaît maintenant fort simple. La complexité d'une chaîne de caractères est la longueur du plus petit programme qui l'engendre (p. ex., le plus petit nombre d'états d'une machine de Turing qui écrit cette chaîne sur son ruban puis s'arrête). Cette complexité est définie à une constante près (comme tout en informatique) dépendante du langage de programmation choisi (du type de machine de Turing choisi).

1.8 De nombreux problèmes opérationnels

Cette théorie fort séduisante n'est pas sans poser de nombreux problèmes opérationnels. En effet, les auteurs démontrèrent que cette mesure de complexité est incalculable, et qu'il est donc impossible de déterminer si un objet a une complexité de Kolmogorov inférieure à une valeur κ donnée. Il existe des nombres proprement gigantesques qui ont une complexité de Kolmogorov ridicule,[16] mais dont des nombres proches numériquement ont des complexités très élevées. Un simple argument de dénombrement démontre d'ailleurs que l'immense majorité des nombres ont une complexité de Kolmogorov maximale, de l'ordre de leur longueur en bits : le seul programme capable de les écrire est la commande `print` suivie de la suite des chiffres du nombre en question.

Par ailleurs, la plupart des problèmes que nous qualifions de "complexes" ont une complexité de Kolmogorov très petite : l'énoncé du théorème de Fermat est d'une simplicité trompeuse sans rapport avec celle de sa preuve et des méchanismes mathématiques en jeu. Les fractales [203, 87, 13] sont un autre exemple de cette situation paradoxale où un programme simplissime[17] peut engendrer des objets d'une complexité que les mathématiciens, physiciens, biologistes et informaticiens sont encore loin de saisir.[18] Il en va de même du jeu de la vie ou des modèles de tas de sable [24, 143]. Ces objets sont donc simples du point de vue de la complexité de Kolmogorov alors que tous les scientifiques s'accordent à les qualifier de complexes. Ce n'est pas le seul paradoxe de la théorie de Kolmogorov.

Cette théorie permet en effet de donner une définition précise, élégante et robuste d'une chaîne de bits aléatoires, c.-à-d. du hasard : il s'agit des chaînes incompressibles, c.-à-d. de complexité maximum (de l'ordre de leur longueur en bits).[19] Or, les résultats de Poincaré et d'Hadamard, rassemblés par la suite sous le terme "théorie du chaos", affirment que l'on peut considérer que le hasard tel que l'on le rencontre dans la nature, pourrait n'être qu'un effet des instabilités intrinsèques de petits systèmes simples déterministes, donc de complexités de Kolmogorov très faibles.

Il me semble que le champ des systèmes complexes est né de ces paradoxes, afin d'essayer de les résoudre : comment comprendre le décalage entre la simplicité de la description d'un processus et la complexité de son comportement face à laquelle nos outils mathématiques se révèlent impuissants ? Il y a fort à parier que c'est plus du côté des outils mathématiques à développer qu'il faut chercher, que du côté de la phénoménologie descriptive, bien qu'il soit probable que les deux approches doivent être menées de front en s'alimentant l'une l'autre (on ne compte plus le nombre de découvertes qui ont été faites à la suite d'expériences "ratées").

2 Une visite guidée du domaine des systèmes complexes

Contrairement à la mathématique,[20] à la physique, à la chimie, à l'informatique, à la biologie, etc., il est difficile de parler *du* système complexe, tant ce champ recouvre des objets, des approches et des pratiques fort différents. Aussi, parle-t-on de systèmes complexes avec un *s*. La recherche d'une unité dans ce champ reste néanmoins une préoccupation fondamentale, car elle sera un premier pas permettant la définition d'une approche globale de ces questions. Les systèmes complexes s'articulent jusqu'à aujourd'hui autour de différents thèmes, dont je vous propose ci-dessous une description aussi rapide qu'incomplète, constituée à partir de mes lectures et de mes rencontres avec les acteurs du domaine.[21]

En mathématique. L'expression "systèmes complexes" est principalement rattachée aux systèmes dynamiques et aux équations aux dérivées partielles, avec, p. ex., le calcul d'exposant de Lyapunov[22] (théorie du chaos) qui permet d'estimer la durée de prédictions fiables sur un modèle, l'étude de la structure algébrique des trajectoires et des attracteurs étranges (en relation avec la théorie des nœuds), l'existence de mesure de Gibbs pour les automates cellulaires (convergence en probabilité que nous explorerons plus en détails au chapitre 2), le calcul de la dimension fractale qui permet de mesurer les objets intermédiaires entre deux dimensions entières, très présents dans la nature, les problèmes multi-échelle.

En physique théorique. "Systèmes complexes" est souvent associé à l'étude des transitions de phases, l'apparition de lois de puissance et de phénomènes sans échelle (les deux sont liés),[23] l'étude des systèmes chaotiques, et surtout plus récemment l'étude des systèmes hors équilibre, de l'auto-organisation et des réseaux dits complexes. L'idée qu'"une fois obtenues les lois (forcément simples) qui régissent la nature on pourra tout expliquer", a fait long feu ; même si on obtient ces lois un jour, il semble maintenant assez incertain que l'on puisse tout prédire pour autant.

Aussi, il s'agit maintenant non pas tant de trouver des modèles *parfaitement fidèles*, que des modèles *traitables*, d'où le succès des modèles d'Ising [172], de percolations avec ou sans amorce [152, 164], des réseaux de neurones et de Hopfield [165], etc., autant d'idéalisations de la nature dont on peut espérer comprendre le fonctionnement pour en déduire des connaissances sur les systèmes réels aux comportements plus chaotiques. C'est ainsi que l'on a découvert un certain nombre de *paradigmes de comportements complexes*.

Comportements critiques auto-organisés. Outre la théorie du chaos que nous avons vu naître précédemment dans les travaux d'Hadamard et de Poincaré, d'autres modèles ont permis récemment d'avancer un peu dans la compréhension de ces phénomènes complexes. Le modèle des tas de sable a permis de révéler que la complexité pouvait émerger dans des processus mécaniques extrêmement simples d'auto-organisation engendrant des paysages énergétiques très structurés, idée popularisée sous le terme *self-organized criticality (SOC)* par Bak, Tang et Wiesenfeld en 1987 [24].[24]

Optimisation multi-critère. Par ailleurs, l'idée que des lois de puissance (et donc le fait que des événements à grande échelle se produisent assez fréquemment)[25] dans les structures construites par les hommes peuvent tout simplement provenir du fait que ces systèmes sont souvent optimisés pour en améliorer la robustesse, idée popularisée sous le terme *heuristically optimized tolerance (HOT)* par Carlson et Doyle en 1999 [59]. Une variante intéressante de cette idée fut introduite par Fabrikant, Koutsoupias et Papadimitriou en 2002 [104] sous le

terme *heuristically optimized trade-off* où l'émergence de loi de puissance serait la résultante d'une optimisation multi-critère algorithmique et gloutonne (coût *vs* qualité de la connexion).[26]

Attachement préférentiel. Une autre idée est le principe d'attachement préférentiel.[27] Il s'agit d'une idée assez ancienne évoquée par Eggengerger et Pólya en 1923 [100] puis étudiée par Yule [332] pour expliquer l'apparition de loi de puissance parmi différentes espèces de fleurs, puis analysée par Simon [290] pour expliquer la distribution des tailles des villes (entre autres), et qui a finalement trouvé son public en 1999 sous la plume d'Albert et Barabasi [32] en écho à la parution de l'article des trois frères Faloutsos de 1999 [106] présentant la première mesure à grande échelle du graphe des routeurs d'Internet, révélant une loi de puissance.[28] En fait cette idée est tellement simple et générique qu'elle est omniprésente et surgit dans à peu près tous les modèles connus (p. ex., dans de simples processus de copies aléatoires [192] ou ceux d'optimisation multi-critère [104, 41]). On peut voir l'omniprésence de ce concept comme une incitation à aller chercher au-delà de cette simple évidence.

Les réseaux d'interaction. On assiste depuis bientôt vingt ans à la mise en réseau des informations personnelles et des affinités via : les réseaux sociaux (FaceBook, etc.), les emails (échanges d'emails entre titulaires de comptes d'une même entreprise, p. ex. Gmail, etc.), les achats et recommandations (Amazon, etc.), les moteurs de recherche (Google, Yahoo, etc.), voire même tous ces facteurs ensemble (Google, p. ex.), les téléphones mobiles (Orange, etc.), les puces RFID (abonnements RATP, etc.), les cartes de fidélité (Carrefour, etc.), la généralisation des équipements GPS, etc. Les bases de données recueillies par ces entreprises permettent d'envisager l'étude des réseaux d'interaction sociale, d'achats ou de transports, à une tout autre échelle qu'auparavant. Ces systèmes d'ailleurs doivent leur succès au consentement passif de l'utilisateur, qui cherche juste à accéder à un service et parfois pour son plus grand bénéfice, p. ex., avec les recommandations et commentaires d'Amazon.

Les données obtenues sont souvent représentées sous forme de graphes divers : hétérogène ou non, bi- ou tri-parti, pondéré ou non, etc. Si l'étude des graphes sociaux a commencé dès les années 1930 en sociologie [221], la multiplication du nombre de sommets est telle maintenant qu'une étude à la main, de visu, n'est plus possible. La représentation même de ces données est devenu un problème de recherche à lui tout seul. Inversement, ce grand nombre de sommets permet aux physiciens d'espérer que les méthodes de physique statistique puissent s'appliquer, d'où l'ouverture de ce champ de recherches nouveau en physique. Newman dresse dans [228] un portrait assez complet de ce que recouvre ce champ de recherches dont les résultats sont encore très rudimentaires, comme il le reconnaît lui-même.

Ces nouvelles mesures ont donc révélé un nouvel "objet naturel" qu'il convient d'étudier : les grands réseaux d'interaction (sociale, biologique, etc.). Selon moi, un des enjeux principaux de ce domaine est l'existence d'une relation entre la *fonction* (à définir) et la *structure* d'un réseau. Comment définir la fonction d'un réseau ? Comment définir et représenter la structure d'un graphe ? Quels paramètres de la structure influent sur la fonction étudiée ? et réciproquement ? Dans quels cas une relation existe-t-elle entre ces fonctionelles ? La contribution de Kleinberg [188] sur le phénomène des "petits-mondes" a été déterminante dans cette direction en créant un lien entre l'algorithmique, la physique et les sciences sociales. Nous verrons au chapitre 1 que cette relation a un sens pour l'étude des graphes petits mondes, mais au chapitre 2 qu'elle apparaît beaucoup moins claire en ce qui concerne les réseaux d'interaction génétique.

Cette problématique est rarement présentée sous l'angle "structure *vs* fonction" alors qu'il me semble être le point fédérateur des travaux portant sur les réseaux d'interaction. Nous

développerons ce thème plus avant dans le chapitre suivant en étudiant le phénomène des "petits-mondes".

En biologie. Le vivant est le domaine par excellence des systèmes complexes. Dans ce domaine, le fondamental et l'expérimental sont étroitement liés : les manipulations qui peuvent s'étaler de plusieurs mois (en biologie) à des centaines voire des centaines de milliers d'années (en écologie), font que le fondamental joue un rôle important dans le développement des techniques de manipulations.

Si la mise en évidence du rôle de l'ADN dans l'hérédité en 1944 par Avery, Mac Leod et Mc Carty [21], puis de sa structure en 1953 par Crick, Franklin et Watson [314, 85] ont permis d'envisager pendant un temps que l'on pourrait bientôt lire en l'homme comme dans un livre, la dure réalité actuelle est que les gènes interagissent de façon extrêmement sophistiquée et que personne ne maîtrise leurs fonctionnements, même pour les organismes les plus simples ne comptant que quelques dizaines de gènes parfaitement cartographiés.

Au caractère fondamentalement multi-échelle (acides aminés $\rightsquigarrow$ protéines $\rightsquigarrow$ noyau $\rightsquigarrow$ membranes $\rightsquigarrow$ cellule $\rightsquigarrow$ différentiation $\rightsquigarrow$ organe, positions relatives $\rightsquigarrow$ corps $\rightsquigarrow$ société, environnement), s'ajoutent les *aspects adaptifs* qui introduisent des boucles de rétroactions difficilement quantifiables car en perpétuelle évolution. Quelques enjeux typiquement systèmes complexes en biologie sont la recherche des gènes et de leurs réseaux d'interaction (l'une des difficultés est de savoir quelles interactions tester, de quelle(s) nature(s), comment les représenter, etc.), comprendre comment passe-t-on d'une échelle à l'autre (des molécules à la cellule, de la cellule à l'organe, etc.), les mécanismes circadiens de synchronisation globale (horloge interne) et plus généralement la morphogénétique (comment passe-t-on du code génétique à la différentiation des cellules qui permettent que telle ou telle forme, avec trous ou sans trou, se développe au bon moment, au bon endroit et à la bonne taille), en passant par le problème du transport des molécules.

À l'évidence, la modélisation joue un rôle central dans ce domaine, moteur à la fois du domaine fondamental et du domaine expérimental : plus que dans toute autre science, le modèle est à la fois le but de l'expérimentation qui produit elle-même un retour de la confrontation du modèle à la réalité. Aussi, tout modèle peut (et doit) être remis en question.

Les systèmes génétiques apparaissent comme des machines extrêmement sophistiquées avec des processus de régulations très intriqués : par exemple, Prochiantz[29] insiste dans ses leçons au collège de France [242] sur le fait qu'il existe certains gènes dont l'expression est indispensable à certaines étapes du développement de l'organisme, au sens qu'en leurs absences, ces développement n'ont pas lieu, mais qui ne sont pas "maîtres" (déclencheurs) pour autant, puisque, si on efface un tel gène chimiquement d'une cellule où il devrait être exprimé, cette cellule arrive en quelques heures à le réexprimer à la bonne concentration. Aussi, Prochiantz conteste l'existence de gène maître et préfère envisager des systèmes de régulation très intriqués avec de nombreux mécanismes de redondance assurant la robustesse nécessaire.

Un cadre naturel pour l'étude de ces interactions me semble être les réseaux booléens que nous explorerons au chapitre 2. Chaque état représente l'expression ou l'inhibition des différents gènes considérés (v. p. ex. [79]), ce qui n'est pas une grosse limitation au regard du fait que l'activation/inhibition d'un gène semble (dans certain cas au moins) prévaloir sur la valeur exacte de sa concentration (qui ne serait qu'une conséquence de son activation/inhibition). Ce modèle est d'autant plus intéressant qu'il possède une expressivité importante tout en permettant d'espérer obtenir formellement des informations sur leurs comportements.

Une particularité des systèmes biologiques qui me semble importante est la *lenteur* des

transitions et transits : la commutation exprimé/inhibé d'un gène prend plus d'une dizaine de minutes et il faut compter environ 3h30 pour qu'un gène parcourt l'équivalent de 16 cellules (v. les leçons du collège de France de Prochiantz [242]). Il est donc essentiel de développer des outils pour étudier ces systèmes *hors-équilibre*. Nous détaillerons au chapitre 2 nos contributions dans cette direction.

En économie. L'économie est également un domaine où les comportements sont hautement adaptatifs et induisent des phénomènes particulièrement compliqués. Selon moi, la spécificité du système économique (marchandises, valeurs, traders et chercheurs, vus comme un tout) est son caractère *"méta-adaptatif"* : il s'adapte non seulement à son état courant mais aux théories faites sur lui-même. Ses acteurs gagnent leur vie sur les fluctuations et il est probable que tout modèle *public* qui rendrait ces fluctuations prévisibles ferait qu'il serait immédiatement invalidé par un changement radical des comportements des acteurs.[30] Par exemple, Frank, Gilovich et Regan montrent dans [123] que la façon d'enseigner l'économie et les modèles proposés affecte considérablement le comportement des acteurs économiques (dans le cas de cette étude, les facultés de coopération).

Une conséquence bien connue de ces comportements adaptatifs sont d'ailleurs les comportements grégaires qu'ils induisent (tous les acteurs vont systématiquement où ils peuvent maximiser leur gain immédiat), qui amplifient de façon très importante la moindre variation du marché. Nous ne sommes d'ailleurs qu'au début de l'exploration des comportements mis en jeu, en particulier l'excellente conférence d'Ariely [17] détaillant comment de très nombreuses forces en opposition et de natures très variées semblent réguler nos comportements, conférence donnée à TED [300] (un site regroupant des conférences de qualité exceptionnelle sur des sujets extrêmement variés, à recommander à toute personne qui a un peu de temps à perdre – ce que je vous souhaite sincèrement).

Indépendamment des aspects adaptatifs, la situation reste délicate puisqu'un résultat récent de Fabrikant, Papadimitriou et Talwar démontre que la question même de déterminer (et donc a fortiori d'atteindre) un équilibre de Nash est un problème déjà PLS-complet pour des cas très simplifiés [105].[31] Ce système est donc probablement constamment *hors équilibre*.

Plusieurs expériences citées par Kirman [184, 185] démontrent dans des cadres bien définis et contrôlés, que malgré des comportements individuels radicalement différents, il émerge un comportement global très homogène des acteurs économiques et qui oscille entre des régimes marqués avec des transitions abruptes et imprévisibles. De façon plus surprenante, on observe les mêmes phénomènes avec des expériences sur les fourmis, et donc que ces comportements pourraient être assez largement indépendant des "aspects humains", ce qui laisse entrevoir la possibilité de construire des modèles relativement simples expliquant ces phénomènes. Nous sommes en tout cas en présence d'un système particulièrement intrigant.

Bien sûr, il manque de nombreux thèmes (p. ex., la vie et l'intelligence artificielles, les séries temporelles, l'astronomie et la géologie, domaines multi-échelle par excellence, etc.), mais j'ai recensé ici les thèmes que j'ai rencontrés dans mes recherches et qui permettent déjà de se faire une idée assez précise de la diversité et des points de convergence des domaines relevant des systèmes complexes.

3 Informatique et systèmes complexes

De façon évidente, l'ordinateur a été et est encore un des outils essentiels à l'essor des systèmes complexes. L'informatique a accompagné le développement du champ des systèmes

complexes depuis ses tout premiers débuts.

3.1 Les relations historiques et leurs évolutions

3.1.1 Simulations

C'est par les ordinateurs que l'on a découvert l'ubiquité de l'émergence de la complexité dans des systèmes que l'on considérait comme "simples" mais que l'on n'avait pas encore pu simuler de façon intensive à la main. Par exemple, l'instabilité des modèles météorologiques, qui a conduit à la découverte du fameux attracteur de Lorenz [202], a été mise en évidence par hasard, en relançant le même programme sur des données intermédiaires sauvegardées tronquées à la quatrième décimale et qui donnèrent un résultat complètement différent du calcul précédent où le calcul avait été mené à son terme avec la précision maximale. Un autre exemple est la première classification intuitive de Wolfram [324] des automates cellulaires élémentaires inspirée par de nombreuses simulations. Ces différents résultats que l'on peut qualifier d'"expérimentaux" ont été à l'origine de nombreux développements théoriques d'une grande richesse, p. ex., la question de l'universalité intrinsèque de la règle **110** [261, 231].

Simulations, régimes transitoires et comportements asymptotiques. Le rapport entre simulations à but exploratoire et théorie n'est cependant pas simple et prouver ce que l'on observe est d'une part loin d'être évident et d'autre part n'apporte pas toujours de réponse satisfaisante. On peut citer par exemple le couac entre les résultats rigoureux de Holroyd [164] affirmant que la densité critique $p_c(n)$ pour l'envahissement par amorce d'une grille $n \times n$ (*bootstrap percolation*) suit la loi $p_c(n) \ln n = \frac{\pi^2}{18} + o(1) \approx 0.54831 + o(1)$ alors que les résultats expérimentaux menés par [10] portant sur des valeurs de n allant jusqu'à 28 800 prévoyaient la valeur 0.245 ± 0.015. Face à ce hiatus expérimentation/théorie, Gravner et Holroyd [149] ont démontré que la convergence vers le seuil est en fait extrêmement lente et qu'il faudrait pouvoir simuler des grilles de côté $n = 10^{20}$ pour commencer à s'approcher de la valeur exacte, et plus précisément $n = 10^{500}$ et 10^{3000} pour l'approcher respectivement à 2% et 1% dans le cas d'une variante du modèle où les calculs sont plus précis. On pourrait y voir comme [26] un échec pur et simple de l'approche expérimentale mais je crois qu'il ne faut pas s'arrêter là et reconnaître aussi l'inadéquation de certains résultats théoriques pour la prédiction de ce qui est observable.

Il me semble évident que l'on ne peut se limiter dans le domaine des systèmes complexes à des approches asymptotiques et qu'il est fondamental d'étudier les régimes intermédiaires de façon à obtenir la meilleure adéquation possible entre théorie et simulations et ainsi contribuer à l'édification de méthodes expérimentales plus sûres, et idéalement, à terme, certifiées. Nous revisiterons cette question dans le chapitre 2 de ce manuscrit.

3.1.2 Collectes et traitements des données

Indépendamment des simultations, l'informatique joue également un rôle capital dans le domaine de la collecte et du traitement des données expérimentales réelles. En effet, grâce à l'informatisation croissante des réseaux sociaux, la possibilité de mener des enquêtes à grande échelle via le web et les blogs, aux interconnexions croissantes des bases de données (avec p. ex., la mise en commun des bases de données sur les réseaux protéiques ou génétiques), ou encore à la généralisation des puces RFID, conduit naturellement à passer de l'étude de quelques dizaines de données à des millions de données. Il se pose alors naturellement la question de la *visualisation* de ces données et de l'*extraction de l'information*. Comme le

souligne Newman dans [228] si l'approche *a manu* et *de visu* était encore possible avec des graphes de quelques dizaines de sommets et d'arêtes, elle est inenvisageable pour les données portant sur des millions voire milliards d'individus dont on dispose actuellement.

Un des tout premiers succès dans ce domaine a été la réalisation de moteurs de recherche efficaces qui ont permis de pouvoir accéder à des informations nettement plus pertinentes par simple mots-clés. Les méthodes développées p. ex. par Google (avec le *page rank* de Brin et Page [56] puis les *hubs and authorities* de Kleinberg [186], calculés itérativement sur la structure du graphe) et Amazon (avec le *latent semantic indexing* pour les recommandations, c.-à-d., la projection des matrices clients/centres d'intérêts sur des matrices beaucoup plus petites, de rang $k = o(n)$ [35, 193, 234, 22]) dans ce domaine ont été particulièrement novatrices et ont marqué définitivement l'entrée de l'algorithmique dans le domaine de l'extraction *structurelle* de l'information.

La collecte, l'analyse, et la représentation des données sur les systèmes complexes reposent de plus en plus sur une expertise informatique. Ces domaines sont devenus des domaines de recherches bien délimités en informatique : p. ex., la conférence annuelle *Graph drawing* en est à sa seizième édition cette année et propose chaque année des défis pour la représentation de graphes issus de domaines variés (sociologie, biologie, ingénierie, etc.). Les questions algorithmiques sous-jacentes sont particulièrement pertinentes et en lien direct avec la problématique de la décomposition de graphes en "régions d'éléments semblables" (par exemple, fortement interconnectés) et les plongements de métrique (domaine actuellement en plein essor tant en théorie des groupes qu'en algorithmique distribuée, pour ne citer qu'eux). On y retrouve également les thématiques difficiles et par nature pluridisciplinaires du recueil d'informations sur les graphes réels : métrologie d'Internet ou des trafics réseaux, mise au point de systèmes de collecte d'informations sur les réseaux sociaux dans le respect de la vie privée, etc. ; de nombreux domaines où se mêlent, technologie, psychologie, interface homme – machine, théorie des jeux, divertissement, marketing, algorithmique, etc.

Algorithmique des grands graphes. Il s'agit en particulier de développer des techniques et algorithmes performants pour traiter, analyser et exploiter les importants flux de données résultants *(data streaming algorithms)*. Il s'agit également de mener un travail de réflexion sur la représentation que l'on donne d'un réseau. Il est banal de dire qu'un graphe peut être représenté de diverses façons : sous la forme classique sommets + arêtes, ou bien biparti sommets – arêtes, ou encore sous la forme d'hypergraphes, voire même d'une chaîne de Markov entre les différents états possibles de l'ensemble du graphe, etc. ; ce que l'on pourra observer/modéliser/analyser facilement dans une représentation ne sera pas nécessairement possible dans une autre.

Prise en compte de la dynamicité. Un des challenges important est aussi la prise en compte de l'aspect *dynamique* de ces structures : p. ex., l'évolution du type des échanges entre protéines au cours de la croissance d'un organisme, l'évolution des connexions Internet, l'influence de la diffusion des nouvelles technologies sur les schémas de connexions, etc., sans parler des aspects géométriques liés aux différentes configurations/conformations possibles d'une même molécule autorisant telle ou telle interaction/réaction à différents instants donnés. Ces aspects dynamiques ne sont encore abordés que très partiellement, et en ce qui concerne l'informatique, principalement via les algorithmiques distribuée, en-ligne, ou de flux, avec toujours une forte composante probabiliste, les systèmes dynamiques discrets (réécriture, automates cellulaires, etc.), et aussi la théorie des files d'attente.

Nous aborderons ce thème à deux reprises : au chapitre 1 avec l'étude du phénomènes des petits mondes, puis au chapitre 3 dans notre étude de l'ordonnancement non clairvoyant.

3.2 Contributions effectives de l'informatique au champ des systèmes complexes

Nous avons vu que l'informatique était aussi l'un des piliers du champ des systèmes complexes avec la formalisation de la notion de calcul, l'ébauche d'une théorie de la complexité algorithmique, de l'information, et du hasard. Je propose de ranger les contributions de l'informatique au champ des systèmes complexes dans quatre catégories : les contributions à l'analyse pratique, les contributions opérationnelles, les contributions fondamentales et les contributions conceptuelles.

3.2.1 Les contributions à l'analyse expérimentale

Ces résultats sont ceux qui permettent de construire les outils qui servent à la collecte, à l'expérimentation, la représentation, la lisibilité, l'exploitation et la compréhension des données. Ceux-ci ont été passés en revue en détail dans la section 3.1.2 de cette chapitre, nous n'y reviendront donc pas ici, sauf pour évoquer cet exemple intéressant de collaboration interdisciplinaire, le travail récent de [52, 78] sur la reconstitution par des moyens informatiques d'un graphe des interactions sociales au Moyen-Âge qui a mis en évidence l'organisation insoupçonnée des communications longue-distance à cette époque.

3.2.2 Les contributions aux techniques d'analyse mathématique

Il s'agit ici des techniques développées pour faciliter l'analyse des processus. On peut regrouper dans cette catégorie les résultats issus des systèmes dynamiques discrets et continus, de l'analyse des processus aléatoires en général (chaînes de Markov, séries temporelles, etc.), de la dynamique symbolique, mais aussi, je crois, la preuve de programme (méthodes de preuve pour l'auto-stabilisation, p. ex.), l'analyse de structures de données, l'algorithmique et les logiques d'automates. Tous ces outils permettent d'obtenir des résultats concrets sur des modélisations des systèmes étudiés. Nous avons vu précédemment que les modèles que l'on pourra analyser complètement sont nécessairement simplificateurs (car sinon ils seraient trop puissants pour être appréhendés), leur analyse n'en est pas simple pour autant. Parmi les modèles les plus étudiés, nous pouvons citer : les modèles d'Ising pour le ferromagnétisme, les modèles de percolation, les modèles de dimères pour les catalyseurs, etc. autant de modèles apparemment simples mais qui ne sont encore compris que très partiellement. Différents domaines de l'informatique ont largement contribué à la meilleure compréhension de ces systèmes.

Techniques d'échantillonnage. Une des questions fondamentale est l'échantillonnage de ces systèmes. Les travaux sur les couplages (ou plutôt auto-couplages de processus) de Jerum, Randall et Sinclair ont été particulièrement importants pour déterminer l'efficacité des méthodes proposées au moins initialement par les physiciens. L'idée générale est d'échantillonner la distribution stationnaire par un processus markovien de mélange par modifications locales successives à partir d'une configuration initiale arbitraire. On démontre que ce processus mélange rapidement en démontrant que deux trajectoires (aléatoires) parties de deux configurations distinctes se rapprochent inexorablement et finissent par se rejoindre au bout d'un temps polynomial (donc rapidement) $T(n)$ connu, en un point qui est alors nécessairement conforme à la distribution stationnaire : il suffit donc de mélanger à partir d'une configuration initiale arbitraire pendant un temps $T(n)$ pour obtenir notre échantillon. Randall présente dans [248, 247] les différentes méthodes proposées pour obtenir ces bornes et les résultats

qui ont considérablement simplifié leurs mises en œuvre comme celle du *path-coupling* où on borne le temps de mélange en étudiant seulement de combien "se rapprochent" en espérance deux configurations qui ne diffèrent qu'en un endroit. À l'inverse, la démonstration de l'existence de goulot d'étranglement dans le "paysage" topologique de la chaîne de Markov associée [248] a permis également de démontrer que certains processus ne mélangent pas rapidement pour certaines valeurs des paramètres (température, etc.). Ces méthodes ont été appliquée avec succès au problème du coloriage aléatoire de graphes, et surtout au pavage aléatoire par des dominos qui modélise, p. ex., le placement de dimères sur des catalyseurs en physique.

Exemples d'interaction entre physique et complexité informatique. Le calcul de la fonction de partition de ces modèles est aussi fort utile car il donne accès à de nombreux paramètres thermodynamiques (entropie, énergie libre, etc.). Il s'agit de calculer le nombre total de configurations possibles (pondérées par leur probabilité). Dans le cas des systèmes monomères-dimères [160], il s'agit essentiellement de compter le nombre de couplages dans un graphe biparti, ce qui se trouve être égal au permanent d'une matrice à coefficients 0 ou 1, or son calcul résistait aux différentes tentatives depuis le début du XIXème siècle sauf dans quelques cas très particuliers [217]. C'est Valiant qui démontra en 1979 qu'en fait le calcul du permanent des matrices à coefficients 0 ou 1 est $\#P$-complet, introduisant par la même la classe $\#P$ des problèmes de dénombrement des solutions aux problèmes NP (plus précisément de dénombrement des chemins acceptants d'une machine de Turing de temps polynomial non-déterministe). La question de l'approximation de sa valeur est resté pendant longtemps une question ouverte très active en algorithmique d'approximation qui fut finalement résolue en 2004 par Jerrum, Sinclair et Vigoda [57, 174, 175] en proposant un schéma d'approximation totalement polynomial (le mieux que l'on puisse espérer pour un problème $\#P$-complet modulo les conjectures habituelles) à base de chaînes de Markov. Cet algorithme de dénombrement repose sur un tirage aléatoire uniforme de couplages parfaits, or cette correspondance s'inscrit dans un résultat plus général qui affirme que pour tout problème auto-réductible, le dénombrement des solutions est polynomialement équivalent au tirage uniforme d'une de ses solutions [176, 138].

Applications des mathématiques discrètes à la chimie. De nombreux résultats en informatique et en mathématique discrète ont des retombées en chimie aussi.

Dénombrement et stabilité des molécules. La démonstration par [183] qu'il existe un nombre exponentiel de couplages parfaits dans les graphes 3-uniformes a permis de démontrer la (méta-)stabilité des fullerènes, molécules organiques composées uniquement de carbones trivalents ayant donc chacun une double liaison à partager avec exactement un de leur trois voisins. Auparavant, seul un petit nombre de ces molécules avaient été démontrées stables avant d'être réalisées effectivement (p. ex., le footballène). Un autre champ d'application de l'algorithmique à la chimie est celui de l'auto-assemblage.

Chimie algorithmique des objets nanoscopiques. Seul un petit nombre de structures nanoscopiques peuvent être obtenues directement par réactions chimiques. Les premières approches pour obtenir une forme particulière proposaient de la construire atome par atome au microscope à effet tunnel, méthode initiée par Eigler et Schweizer pour IBM en 1990 [101]. La construction est alors très lente et ne permet pas de construire un grand nombre d'objets.

Les travaux algorithmiques de Winfree ont permis de proposer une autre méthode plus efficace reposant sur la construction de tuiles élémentaires constituées d'un brin d'ADN replié

sur lui-même en carré (que l'on peut multiplier exponentiellement très facilement par réaction en chaîne par polymérase (PCR)) qui vont se coller ensemble spontanément, une fois en solution, pour former la structure désirée [321, 322, 296, 275]. Ces méthodes s'inspirent des techniques développées en algorithmique des pavages par tuiles et des automates cellulaires tout en ayant leur propre spécificité.

Nous aborderons ce sujet de l'auto-assemblage en profondeur au chapitre 4 où nous étudions plus précisément les relations entre temps et ordre de l'assemblage pour la conception de schémas efficaces d'auto-assemblage.

D'autres méthodes algorithmiques ont été également conçues par cette équipe : il s'agit de partir d'un long ADN de virus et de le replier en la forme voulue à l'aide d'"agrafes", de courts brins d'ADN dont les formules ont été déterminées par ordinateur en résolvant des problèmes d'optimisation [272] ; bien que les calculs des agrafes doivent être reconduits de zéro pour chaque taille désirée de la structure souhaitée (ce qui est une forte limitation de la méthode au niveau conceptuel), tous ses brins (virus et agrafes) peuvent être répliqués exponentiellement par PCR et la méthode permet bien de produire ces assemblages "en masse".

Réseaux d'interaction en biologie. L'informatique joue aussi un rôle de plus en plus important en biologie. La théorie des réseaux d'automates booléens a permis de proposer des explications sur les relations entre le nombre de spécialisations possibles des cellules en fonction du nombre de gènes en interactions dans le réseau génique associé [79] : les états `true`/`false` correspondent aux états activé/inhibé des gènes et les fonctions booléennes aux relations d'interaction entre eux (activation/répression). Les réseaux d'automates booléens sont encore très mal compris et leur analyse, tout particulièrement de leur fonctionnement hors-équilibre, pourrait conduire à des avancées significatives dans notre compréhension des systèmes génétiques.

Nous aborderons ce sujet en profondeur au chapitre 2 où nous proposons une analyse (très partielle) d'un modèle ultra-simplifié de ces réseaux : les automates cellulaires probabilistes. Nous nous intéresserons tout particulièrement à leur régimes transitoires et évoquerons en détail les difficultés soulevées par ces analyses.

3.2.3 Les contributions conceptuelles

Cette catégorie réunit les travaux des autres disciplines qui permettent d'apporter un éclairage nouveau sur le champ des systèmes complexes. Il s'agit de résultats originaux, propres à un domaine, qui, parce qu'ils sont surprenants, contre-intuitifs élargissent le champ du possible en levant des a priori erronés qui pourrait conduire à négliger ou mésestimer telle ou telle possibilité dans la modélisation d'un système.

La randomisation. L'algorithmique randomisée[32] est pour moi la contribution principale de l'informatique à cette catégorie. *Elle a révélé à quel point nos intuitions sont fausses sur ce que l'on peut faire avec très peu de moyen, pour peu que l'on ait accès à une source d'aléatoire.* La recherche de processus efficaces et économes a en effet permis d'obtenir une série de résultats qui ont *changé notre vision* de la complexité dans de nombreux domaines. Parmi ceux-ci, nous pouvons citer : les compteurs probabilistes qui ont révélé que $\log\log n + O(1)$ bits suffisent pour incrémenter jusqu'à n un compteur probabiliste avec une précision relative arbitraire [222, 115], alors que l'intuition nous conduirait à penser que $\Omega(\log n)$ bits sont nécessaires ; la *zero-knowledge* qui a démontré qu'il était *possible* de convaincre que l'on possède une information sans *rien* en révéler [141] ; l'auto-correction de programmes introduite par [47, 313] dont l'idée est qu'introduire de l'aléatoire dans les entrées d'un programme bugué sur

une part inconnue de $x\%$ des entrées, permet d'obtenir une réponse juste sur *chaque* entrée avec une probabilité de $(100 - O(x))\%$.

Plus généralement, l'algorithmique randomisée et la complexité aléatoire ont démontré qu'il existait des méthodes pour résoudre de façon simple et efficace des problèmes dont les solutions déterministes sont pourtant très sophistiquées voire compliquées. Il est certain que nous devons tenir compte de ces résultats lorsqu'il s'agit de modéliser la nature et ce pour plusieurs raisons : tout d'abord, ses algorithmes sont économes (en temps et ressource) et surtout bien plus "naturels" au sens qu'ils s'expliquent souvent facilement, et de façon concise ; d'autre part, ces algorithmes étant bien plus simples, parcimonieux, et efficaces que leurs alternatives déterministes, ils sont donc favorisés par les processus de sélection naturelle : p. ex., le gap exponentiel entre compteurs déterministe et aléatoire fait que le second a bien plus de chance d'apparaître "par hasard" que le premier, d'autant plus que les algorithmes probabilistes sont souvent plus robustes aux erreurs.

La connaissance de ces résultats permet p. ex. une analyse surprenante mais pertinente du recours aux oracles (mises à mort d'animaux divers et variés, etc.) dans les sociétés plus anciennes : partant du fait que le théorème de von Neumann assure qu'une stratégie optimale en théorie de jeux [310] se doit d'être aléatoire, Ruelle [276] conclut que le recours aux oracles a sans doute été pendant longtemps une façon d'injecter un hasard salvateur dans les décisions politiques importantes et qu'il est probable que les peuples y ayant recours ont fini par gagner sur ceux dont les décisions étaient purement déterministes et donc défavorables.

Algorithmique distribuée et cryptographie. Deux autres thèmes me semblent porter aussi des contributions importantes au champ des systèmes complexes : l'informatique distribuée et la cryptographie, deux domaines qui se sont d'ailleurs développés en interactions permanentes avec l'algorithmique randomisée : l'informatique distribuée, car elle s'intéresse à l'étude de l'émergence de propriétés globales par simples interactions locales (élection d'un chef,[33] robustesse et auto-stabilisation, consensus, synchronisation, réseau pair-à-pair, c.-à-d. sans chef,[34] etc.) ; la cryptographie, car en se posant la question de la complexité du décryptage, étudie comment générer suffisamment de complexité de façon économique. Cela ne surprendra donc personne que l'effet petit-monde soit devenu très vite un sujet de première importance au sein de la communauté algorithmique distribuée dès que sa nature algorithmique a été mise en lumière par Kleinberg [188]. Les retombées de l'étude de ce phénomène ont été, entre autres, la conception de nouveaux protocoles randomisés plus efficaces pour les réseaux pair-à-pair, mais nous y reviendrons plus longuement dans le chapitre suivant.

Nécessité de la transmission de ces concepts. On observe plus facilement un phénomène ou un processus qui nous est familier : on ne lira vraisemblablement pas dans un graphe d'interaction moléculaire un compteur probabiliste sans savoir ce que c'est. On ne lance jamais (ou presque) des expériences au hasard, mais en ayant toujours une idée préconçue au départ. La réflexion informatique sur la façon dont peut être traitée l'information pourrait donc permettre d'élargir le champ des expérimentations in vivo. Je crois qu'il est très important d'informer les modélisateurs des contributions de l'algorithmique randomisée, à mon avis essentielles mais externes au champ des systèmes complexes. Cela signifie un travail important d'extraction et de clarification de ces idées souvent touffues et exprimées initialement dans le langage propre à la discipline dont ils sont issus.

C'est pourquoi j'ai consacré une part importante de mon temps à l'enseignement de ces

[33] *Hug !*
[34] ~~*Hug !*~~

résultats que je juge fondamentaux à des publics non-informaticiens, lors des conférences ou des différentes écoles où j'ai été invité à participer : International summer school on complex systems 2007, Instituto de systemas complejos de Valparaíso ; International summer school on discrete mathematics 2008, Instituto de systemas complejos de Valparaíso ; Science en fête 2003, à l'École normale supérieure de Lyon ; Article de presse « Thèse idiote : le hasard fabrique des certitudes » dans le journal *Le minotaure* n°1, avril-juin 2003 [279].

3.2.4 Les contributions fondamentales

Il s'agit des résultats qui pourraient contribuer à la fondation d'une science des systèmes complexes. Ils portent sur la caractérisation de la complexité, les moyens de la mesurer, et l'obtention de classifications de systèmes en fonction de leur comportement. Comme nous l'avons vu précédemment, l'informatique est une science de la complexité dont elle a contribué à définir un cadre permettant de l'appréhender de manière quantitative. Les questions de l'importance de l'influence du hasard, de la classification entre calculs rapides (régimes transitoires brefs) et calculs exponentiels (états métastables appelés à durer donc perceptibles), et des limites entre indécidabilité (ou du moins efficacité de la vérification) et expressivité des langages, sont actuellement au cœur des débats de la communauté informatique, tant en complexité algorithmique qu'en conception des langages de programmation et preuves de programmes. Des progrès spectaculaires ont été accomplis récemment en informatique dans cette direction en quatre étapes principalement.

Utiliser des problèmes difficiles pour engendrer du véritable hasard déterministe. C'est la cryptographie qui a accompli les premières étapes. En renonçant à l'inviolabilité *absolue* (face à un adversaire arbitrairement puissant) telle qu'elle a été définie par Shanon [289] et qui implique de partager une clé aléatoire de la même longueur que le message à transmettre, Shamir puis Blum et Micali [287, 48] ont proposé de se contenter de suites aléatoires polynomialement *imprévisibles*, c.-à-d. telles que pour tout i, aucun circuit de taille polynomiale ne puisse faire une prédiction correcte du $(i+1)$-ème bit à partir des i premiers avec une probabilité significativement meilleure que $\frac{1}{2}$ (précisément, $\frac{1}{2} + O(\frac{1}{\text{longueur de la suite}})$). L'existence de telles suites reposent sur la conjecture de l'existence d'une fonction *à sens unique (one way)*, c.-à-d. facile à calculer (en temps polynomial) mais difficile à inverser [139, 140, 168, 159, 18] : p. ex., le produit de deux nombres premiers de même taille. Ceci a permis la transmission sûre de fichiers de grande taille (en réduisant polynomialement le nombre de bits aléatoires nécessaires pour garantir le secret face à un adversaire) et surtout de bâtir des protocoles non plus sur des astuces mais sur des conjectures largement admises dans la communauté scientifique qui leur ont conféré une résistance inégalée aux attaques tant des hackers que des mathématiciens (p. ex., le protocole RSA [262, 271] grâce auquel le commerce par Internet et les communications sans fil se sont développés).

Hasard déterministe $\Leftrightarrow \exists$ fonction vraiment difficile à calculer. Par la suite, Yao [330] a établi en 1982 que la notion de suite polynomialement imprévisible correspond en fait à la notion de suite *pseudoaléatoire*, c.-à-d. telle que tout circuit randomisé polynomial alimenté par cette suite produit les mêmes résultats avec des probabilités asymptotiquement identiques que lorsqu'il est alimenté par une suite réellement aléatoire de même longueur. Motivés par la résolution de la conjecture $P = BPP$, Nisan et Widgerson puis avec Impagliazzo et Umans [229, 169, 170, 307] en conclurent qu'un nombre logarithmique de tels bits suffit pour peu qu'il existe bien dans $EXP = \text{DTIME}(2^{n^{O(1)}})$ une fonction calculable seulement par des circuits de taille au moins $\Omega(2^{\varepsilon n})$ pour une certaine constante $\varepsilon > 0$. Une telle borne

inférieure en taille de circuit impliquerait donc directement que $P = BPP$. Enfin, Kabanets, Impagliazzo et Wigderson [167, 179] montrèrent que l'existence de suite pseudoaléatoire est en fait *équivalente* à l'obtention de borne inférieure de taille de circuits.[35]

Hasard et complexité. Il s'en suit donc que l'existence de suites pseudoaléatoires, c.-à-d. du *hasard déterministe*, est *intimement liée à des questions fondamentales posées par la théorie de la complexité* en informatique depuis sa création. Au delà de la thématique du hasard, ces résultats démontrent de façon assez surprenante (je trouve) que la possibilité pour un système simple (polynomial) de générer des motifs très compliqués en apparence (*i.e.*, qui paraissent aléatoires à tout calculateur polynomial) dépend de l'existence de fonctions calculables en temps exponentiel qui nécessitent effectivement des circuits de taille exponentielle. Tout progrès significatif dans cette direction aura donc des conséquences importantes sur les fondements d'une véritable science des systèmes complexes et en particulier sur la compréhension de l'émergence de structure complexes au point de paraître aléatoires dans des systèmes déterministe d'apparence pourtant simple (p. ex., *one-way*). Les liens entre ces résultats en théorie de la complexité et les systèmes complexes restent à construire, nous y reviendrons dans la conclusion de ce mémoire.

3.3 Le rôle central de la modélisation

L'un des objectifs premiers du champ des systèmes complexes est de comprendre l'émergence, la fabrication, l'organisation de la complexité dans la nature. Aussi, une des activités principales, sinon le cœur du domaine, est la conception et l'étude de modèles de la nature. Comme nous l'avons vu précédemment, la modélisation doit faire face à l'indécidabilité : un modèle trop proche de la réalité aura quasiment inévitablement la puissance Turing et donc la plupart des propriétés qui le concernent seront indécidables ; inversement, trop simple, il ne nous apprendra rien sur le phénomène étudié. L'art de la modélisation consiste donc à résoudre le compromis entre réalisme, portabilité, expressivité, généricité et accessibilité ("prouvabilité") du modèle.

Il ne faut donc pas être surpris par l'apparente simplicité des modèles que l'on peut rencontrer dans ce domaine : ils sont la conséquence des impossibilités théoriques mentionnées précédemment. La difficulté de la modélisation est donc de concevoir des modèles qui capturent au moins une partie des phénomènes observés et nous permettent d'en apprendre plus sur ce phénomène, par exemple de valider une théorie ou au contraire d'explorer ou de révéler sa possible nature. Comme le souligne Prochiantz [242],[36] il n'est pas vraiment important qu'un modèle soit juste ou non, l'important est, s'il faux, qu'il nous apprenne quelque chose sur le phénomène étudié ou nous pousse à découvrir d'autre chose. En particulier, il souligne le rôle du modélisateur qui ne doit pas être dogmatique au point de se leurrer à imposer son modèle plutôt qu'un autre par pur parti pris. L'évolution des modèles des petits-mondes de Watt-Strogatz à nos jours, que nous verrons au chapitre 1, est particulièrement édifiante à ce propos. Une des difficultés est qu'un modèle, même meilleur, ne correspond pas toujours à l'usage que l'on souhaite en faire. D'où la coexistence de différents modèles sensés modéliser les mêmes phénomènes, mais servant à en traiter des parties si différentes qu'il est parfois difficile d'en retrouver le point commun.

La modélisation est également à la base de notre discipline. En effet, l'informatique s'est fondée sur sa capacité à proposer des formulations mathématiques à des problèmes (p. ex., d'optimisation) et d'en faire la théorie de la résolution. Les résultats présentés dans ce manuscrit sont tous basés sur des modèles au travers desquels nous espérons apprendre des choses sur les phénomènes correspondants dans les situations réelles. Dans le chapitre 1, nous verrons comment nous avons modifié le modèle algorithmique des petits-mondes proposé par

Kleinberg afin d'en augmenter la généricité puis de lui donner un certain réalisme en démontrant que l'on peut l'engendrer avec très peu de mémoire et en très peu de temps. Nous verrons au chapitre 2, les analyses que l'on peut espérer faire d'un modèle ultra-simplifié des réseaux de gènes, les automates cellulaires probabilistes. Au chapitre 3, nous verrons comment il est possible de proposer un cadre permettant d'analyser des algorithmes en l'absence de toute connaissance de l'instance qu'ils doivent traiter et de l'avancement de ce traitement. Enfin au chapitre 4, nous présenterons un modèle algorithmique qui a permis à Winfree en 1998 de fonder un nouveau procédé de construction d'objets nanoscopiques, réalisés avec succès par la suite en utilisant des brins d'ADN [274, 275].

4 Peut-on parler d'une science des systèmes complexes ?

Il est évident que les systèmes complexes sont une thématique scientifique au sens où il s'agit d'étudier des phénomènes déterminés par des méthodes scientifiques (reproductibles, etc.). Cependant, il me semble prématuré d'en parler comme d'une science à part entière car *elle n'a pas produit pour l'instant d'outils ou de méthodes qui la distinguent des autres champs*.

On peut dire que la mathématique s'est distinguée des autres sciences par sa capacité à affirmer des propriétés certaines sur des objets abstraits (et en particulier à démontrer des résultats d'impossibilité certaine) ; de même la physique s'est distinguée en proposant une démarche à la fois expérimentale et théorique permettant de prédire les phénomènes naturels ; et l'informatique s'est distinguée en définissant la notion de calcul, ce qui a permis de quantifier pour la première fois le temps nécessaire pour résoudre un problème en fonction de la taille de son entrée ; etc. On ne peut cependant pas encore affirmer que les systèmes complexes se soient détachés des autres disciplines et le développement de leur théorisation dépend encore beaucoup trop des autres sciences.

Quelques avancées importantes que l'on peut espérer seraient la mise en évidence d'outils qui permettent de "gérer" la complexité, par exemple, la découverte d'un procédé mathématique qui permettrait d'énumérer un grand nombre de cas de façon concise, ou bien la découverte d'un critère de complexité qui permette de classifier des systèmes dynamiques en fonction de leur comportement, etc. Il me semble que c'est dans cette direction "*coping with complexity*" (affronter la complexité) que les systèmes complexes s'imposeront en tant que science à part entière.

Contrairement à Wolfram dans [325], je ne crois pas qu'il faille s'abstraire de la mathématique mais au contraire s'appuyer dessus, ainsi que sur la physique et la biologie, pour développer les nouveaux outils dont nous avons besoin, de la même façon que l'informatique s'est appuyée sur la mathématique (et s'y appuie toujours !) pour développer la complexité algorithmique qui lui a permis de prendre son autonomie vis-à-vis de la mathématique en créant la science du calcul.

5 Interactions locales et propriétés globales émergentes

Nous étudierons plus précisément dans les chapitres qui suivent, différents aspects de l'émergence de lois globales sous l'effet d'interactions locales. Ce thème était celui de l'ATIP CNRS jeune chercheur Blocoglobo (qui a financé mes recherches et celles de mes étu-

diant(e)s de 2005 à 2006) et guide l'ensemble de mes travaux depuis 2002. Nous aborderons cette question sous l'angle algorithmique.

- Le chapitre 1 étudie l'émergence d'une propriété globale : le fait que les gens ne soient qu'à quelques poignées de mains les uns des autres, bien que leurs relations amicales, professionnelles ou autres, ne soient (en général) pas choisies dans ce but. Nous étudierons plus particulièrement les relations entre la structure des contacts longue-distance dans les réseaux sociaux et la topologie sous-jacente du réseau en proposant une extension du modèle introduit par Kleinberg dans [188] et de son algorithme de routage. Nous démontrerons la plausibilité de ce modèle en en proposant le premier schéma entièrement distribué, ne requérant que des interactions locales extrêmement limitées (polylogarithmiques) en temps comme en espace en regard de la taille du réseau pour émerger d'un réseau quelconque. Ces résultats [195, 196, 88, 89, 90, 194, 197] pourraient avoir des applications intéressantes pour l'optimisation des réseaux pair-à-pair. Ce chapitre aborde sur un exemple la question générale de l'influence de la fonction d'un réseau sur sa structure et réciproquement.
- Le chapitre 2 étudie un des modèles par excellence des interactions locales : les automates cellulaires. Nous nous intéresserons tout particulièrement à leurs régimes asynchrones probabilistes. L'objectif de ce chapitre est de tenter de recenser les différents comportements possibles des petits automates, c.-à-d. en se limitant à ceux (déjà nombreux !) ayant peu d'états (deux) et des relations de voisinages à courte distance en 1D et en 2D. Si certains automates présentent des comportements relativement faciles à analyser, nous avons été très surpris par la richesse des comportements d'automates dont la description est aussi "simple". Certains automates 1D présentent des transitions de phases très marquées sur lesquelles nous ne disposons que de bornes très lâches. Nous étudierons en particulier l'émergence de motifs dominants et les structures qu'ils imposent sur les diagrammes espace-temps. Nous avons pour cela développé des outils spécifiques, permettant en particulier de mener à bien de manière lisible et aisément vérifiable les études de cas inévitables [110, 111, 107, 112, 255, 256, 258, 257, 254].
- Le chapitre 3 étudie le cas d'algorithmes prenant des décisions basées sur une vision extrêmement localisée dans le temps mais dont l'auto-stabilisation permet d'obtenir des garanties sur leur coût global. Nos travaux se basent sur le modèle des algorithmes non-clairvoyants définis par Edmonds dans [93]. Ces algorithmes ignorent tout de l'efficacité de ce qu'il font, de leur progression, et de ce qu'ils auront à faire, et ne peuvent donc prendre que des décisions locales basées sur le présent. Le cadre défini par Edmonds permet d'envisager d'une part des algorithmes que l'on peut effectivement implémenter en pratique (car ils ne présupposent rien sur les instances qu'ils doivent traiter) et, d'autre part, dont les performances prouvées sont réellement les performances observées en pratique. Nous verrons qu'il est possible de proposer des algorithmes compétitifs dans des cadres plus généraux que ceux étudiés par Edmonds, incluant des dépendances arbitraires et inconnues entre les tâches [265, 264, 266, 263].
- Le chapitre 4 étudie les conséquences des règles d'auto-assemblage local sur la géométrie globale des objets ainsi construits. Notre travail est basé sur le modèle d'auto-assemblage algorithmique proposé par Winfree [321] et réalisé effectivement dans [275] pour contruire des objets nanoscopiques à partir de briques constituées de brins d'ADN astucieusement repliés sur eux-même. Nous étudierons plus particulièrement

l'influence de l'ordre local de l'assemblage sur les constructions possibles et sur leur temps d'assemblage [39].

J'ai souhaité présenter nos travaux à l'intérieur de chaque thème dans un cadre unifié. La rédaction de ce manuscrit a été l'occasion de mettre ces travaux en perspective en y incorporant de nombreuses remarques utiles qui ne trouvent pas toujours aisément leur place dans des publications traditionnelles (comme des explications sur les raisons d'un échec). Dans certains cas, cette prise de recul a permis d'obtenir des résultats nouveaux à la fois plus simples dans leur formulation et meilleurs dans leur performance (comme au chapitre 1). Ce manuscrit présente aussi des analyses expérimentales complémentaires qui permettent de mieux cerner les phénomènes que nous ne comprenons pas encore analytiquement (v. chapitre 2). En général, nous ne présenterons que les grandes lignes des preuves de nos résultats et renverrons aux articles correspondants pour leur présentation détaillée. Ce document se conclut avec le chapitre 5 où j'évoque les différentes directions de recherches que je souhaite développer à l'avenir ainsi que quelques grands axes dans lesquels l'informatique pourrait contribuer pour aider à l'élaboration du champ des systèmes complexes.

Notes du chapitre

[2]Se référer à [161] pour l'histoire amusante de ce résultat, où sous la pression du calendrier Poincaré faillit publier un résultat contraire (donc faux) en 1888, puis, grâce à la vigilance d'un relecteur, revit sa copie et dut payer de ses deniers l'édition du résultat correct en 1890.

[3]Ce problème est toujours d'actualité et de nouvelles trajectoires stables aux problèmes à n corps, en forme de huit, de nœuds variés, et même à symmétries tridimensionnelles (et donc non-planaires) sont découvertes régulièrement [162, 219, 227, 220].

[4]Les chapitres 21 à 23 consacrés à l'informatique sont malgré tout un peu confus, ce qui n'ôte cependant rien aux bénéfices et au plaisir que l'on tire de cette lecture que je recommande chaudement.

[5]J'ai toujours trouvé la dénomination *computer science* inadéquate car l'informatique est avant tout la science du calcul et non des ordinateurs : l'informatique survivra naturellement aux ordinateurs qui ne sont qu'un support parmi d'autres pour faire du calcul et qui seront vite oubliés dès que l'on aura trouvé mieux. Il est d'ailleurs instructif de consulter la page wikipedia relatant l'histoire du choix du mot *computer science* [317]. En effet cette appellation n'a été choisie par la communauté américaine que par dépit, après qu'une entreprise qui avait déposé le nom *Informatics* au registre du commerce ait poursuivi systématiquement toute université qui créait un département portant ce nom.

[6]Ce qui démontre que certains faits initialement ardus finissent par percoler jusqu'au grand public.

[7]En fait, le mal est plus profond : il est impossible de prouver qu'une théorie est consistante (cohérente) au sein de cette même théorie, c'est le second théorème d'incomplétude de Gödel [137].

[8]Contrairement à la numérisation, la notion de problèmes indécidables a en revanche beaucoup de mal à percoler, même en mathématique.

[9]"Arbitrairement grand" ne signifie pas "infini", mais que l'on peut lire et écrire aussi loin que l'on veut. Cette différence subtile mais essentielle assure qu'il n'existe qu'un nombre dénombrable de configurations possibles, supposer le contraire changerait radicalement la puissance du modèle, p. ex., on ne saurait pas comment (d)écrire la configuration initiale.

[10]Une machine de Turing peut avoir plusieurs têtes de lecture/écriture, plusieurs rubans, etc. mais on peut démontrer que tous ces modèles sont équivalents au premier. C'est cette robustesse du modèle et sa capacité à simuler les preuves mathématiques et les programmes informatiques, qui est à l'origine de la thèse de Church-Turing.

[11]Cette thèse pourrait bien sûr être un jour contredite par la découverte d'un nouveau modèle de calcul plus puissant, mais cela impliquera de passer d'abord par une révolution en mathématique consistant à écrire des preuves mathématiques de manière *radicalement* différente de ce que nous faisons actuellement, ce qui paraît pour le moment bien difficile à imaginer. Remarquons que si les ordinateurs quantiques permettent d'envisager une accélération de certains calculs (la recherche dans une liste non-triée ou factorisation des grands nombres entiers, p. ex.), ils peuvent être simulés par une machine de Turing et ne sont pas plus puissants en terme de *modèles de calcul*.

[12]Un *automate cellulaire* est une version parallèle de la machine de Turing (nous le verrons en détails plus tard), constituée d'un ruban bi-infini où la tête de lecture/écriture est répliquée à l'identique en face de toutes les cases du ruban et où l'état de chaque tête est la lettre écrite sur sa case du ruban. Un automate cellulaire est *captif* si lorsqu'elle change d'état, chaque tête prend son nouvel état parmi son état actuel et celui de ses deux voisines. Cette restriction permet de manière paradoxale d'assurer que la probabilité que l'automate soit universel tend vers 1 quand le nombre d'états de la tête répliquée tend vers l'infini.

[13]Un autre exemple de système de calcul universel de complexité apparente très modeste est la *correspondance de Post* [241, 218] : étant donnés des dominos sur lesquels sont écrits deux mots d'un alphabet fini, l'un au dessus de l'autre, et tels qu'il n'existe qu'un nombre fini de dominos distincts ; vous disposez d'une réserve arbitrairement grande de chaque type de dominos ; la question est de déterminer si vous pouvez aligner des dominos de sorte que le mot écrit au-dessus soit le même que celui écrit au-dessous. Treize dominos suffisent pour simuler tous les calculs possibles.

[14]La preuve de programme est une activité essentielle de notre domaine : un algorithme juste mais mal implémenté est inutilisable, voire même dangereux suivant le degré de confiance de ses utilisateurs.

[15]L'idée préconçue que programme et donnée sont deux objets bien distincts a sans doute permis le succès il y a quelques années du mythe très vendeur prétendant qu'une fois l'ADN "décodé", on aura les moyens de soigner toutes les maladies. Il s'agit là d'un jeu très dangereux car cette publicité excessive a servi à convaincre le grand public du bienfait de l'approche génétique ; et maintenant qu'il faut bien reconnaître que l'on est loin de pouvoir réaliser les prouesses promises, d'autres

surfent sur cette vague positive pour promouvoir des utilisations bien plus discutables de l'ADN, et le grand public nage dans la plus grande confusion.

[16]Une machine de Turing à douze états suffit pour écrire sans erreur tous les chiffres (et uniquement ceux-ci) du nombre impressionnant :

$$\underbrace{1111111111111111\cdots1111111111}_{\text{longueur : } 6\times 4096^{4096^{\cdot^{\cdot^{\cdot^{4096^4}}}}}\ \updownarrow\ 4096 \text{ apparaît } 166 \text{ fois}}$$

Pour plus d'informations, reportez-vous au problème dit du "castor affairé" [209].

[17]FractaleDeMandelbrot(n: $\mathbb{N}$):
for all $c \in [-2,1]\times[-1,1]$ **do**
let $f_c = z \mapsto (z^2+c)$ **in**
if $|f_c^n(0)| \leqslant 2$ **then** paint c in noir.

[18]Remarquez enfin que l'appartenance à l'ensemble limite de Mandelbrot $\{c : \forall n, |f_c^n(0)| \leqslant 2\}$ est indécidable par une $\mathbb{C}$-machine de Turing [18, Chap. 16].

[19]Cette définition fut généralisée aux suites infinies par Martin-Löf [208] qui formalisa par la suite la notion de tests statistiques et démontra que les chaînes incompressibles passent correctement ces tests ; lire le passionnant dialogue orchestré par Ferbus-Zanda et Grigorieff sur cette question [124].

[20]L'usage de mathématique au singulier est récent et témoigne d'une volonté de faire apparaître clairement l'unité de cette science (la preuve de faits certains dans un cadre abstrait précis) et de surmonter les particularités de chacune de ses spécialités qui n'étaient qu'apparentes ou historiques.

[21]Cette visite est donc biaisée par mon expérience, d'où le titre "visite *guidée*" de cette section.

[22]Un système dynamique a un exposant de Lyapunov $1/\tau$ si deux trajectoires s'éloignent dans un rapport de l'ordre de $e^{t/\tau}$ en fonction du temps t. Par exemple, pour les modèles actuels de la météorologie τ est estimé à quelques jours ; pour le modèle gravitationnel, τ est estimé à plusieurs centaines d'années avec la précision des mesures actuelles (p. ex., on ne sait pas si Pluton sera encore dans le système solaire dans cinq cents ans), ce qui explique la bonne stabilité des prédictions des trajectoires dans notre système solaire.

[23] Il existe une relation très étroite entre lois de puissance et phénomènes sans échelle. Les quantités ayant une unité en physique, pour qu'il existe une relation entre des quantités différentes, il faut pouvoir annuler les unités. Cela peut se faire de deux façons : soit il existe une valeur particulière X_0 du paramétre X et alors diviser X par X_0 annulera son unité et n'importe quelle relation en $\frac{X}{X_0}$ est possible, mais alors la valeur de référence X_0 définit une échelle sur X ; soit il existe une relation nécessairement polynomiale entre les différents paramètres $X, Y, \ldots$ permettant d'annuler les unités, d'où les lois de puissance. Une autre façon de voir est que les seules distributions f telles que pour tout $\lambda > 0$, il existe $\mu > 0$ telles que $f(\lambda X) = \mu f(X)$ (c.-à-d., étirables sans changement notoire) sont les distributions polynomiales.

[24]Dans ce modèle, si une colonne de grains de sable dépasse une certaine hauteur, les grains en excès tombent sur les côtés, ainsi de suite. L'ajout répété de grains de sable au centre d'un carré produit des motifs d'une majesté qui nous échappe encore malgré de nombreux efforts [143].

[25]p. ex., l'existence de villes de très grande taille, de routes très longues, de salaires très élevés, etc.

[26]Cet article a laissé un goût amer à leurs auteurs car ce qu'ils y démontrent (une loi à queue lourde à paramètre fixé) n'est pas ce qu'ils énoncent (une loi de puissance pour tout paramètre fonction du temps), ce qui fut éclairci par la suite par Berger et coll. [41] qui démontrent que ce modèle ne produit effectivement pas une loi de puissance au sens strict.

[27]Dans un réseau, le nouvel arrivant se lie aux nœuds déjà présents avec une probabilité croissante avec leurs degrés courants (p. ex., linéairement). L'approximation du champ moyen permet alors de prédire très facilement une loi polynomiale sur les degrés, la preuve exacte est autrement plus ardue, v. le survey [50].

[28]L'existence même de cette loi de puissance est toujours en débat actuellement et la possibilité d'un biais lié à l'instrument de mesure est toujours en suspens [69, 8].

[29]A. Prochiantz (Cours n°3 au Collège de France, 2007, 31ème min 17 sec) : *"Il y a une chose que je déteste c'est le concept de gène maître. Vous savez on dit toujours il y a des réseaux génétiques et puis il y a un gène qui est maître donc ça veut dire que c'est lui qui va décider. Ça n'existe pas les gènes maîtres donc la génétique c'est toujours une centaine de gènes qui fonctionnent ensemble et quand il y en a un qui déborde, il y a les autres qui le ramènent à la raison, c'est-à-dire qu'il y a rarement des variations de plus de 5 à 10% dans l'expression d'un gène parce que ça fonctionne en réseau. Il y a une sorte de régulon-multi-génique... Mais il y a des cas où on peut dire qu'il y a des gènes qui sont particulièrement plus maîtres que les autres ou moins maîtres que les autres et paxis c'est un cas comme ça"*.

[30]Il est d'ailleurs intéressant de noter qu'il existe sans doute des modèles assez performants mais gardés *secrets* de l'économie ; p. ex., le fond d'investissement américain *Renaissance Technologies* qui recrute exclusivement des mathématiciens universitaires a réalisé

l'une des plus grosse fortune en bourse en 2007 sans que personne ne puisse expliquer comment ils ont procédé : la réponse à la question de R. Engle (prix Nobel d'économie) lors d'une conférence à l'université de New York par son fondateur, J. Simons, lui même ex-mathématicien dont le revenu (notre future impact factor, autant commencer dès maintenant à apprendre de nos futurs maîtres) est estimé actuellement à plus d'un milliard de dollars américains par an, est un rien provocatrice : « la "vérité" en matière de finance change en permanence. Il serait nécessaire de bâtir une nouvelle théorie presque chaque semaine » [305].

[31] Un problème est PLS-*complet* si le résoudre efficacement permet de calculer avec la même efficacité tout objet dont l'existence est prouvée par la décroissance d'une fonction potentielle [105]. Ceci implique qu'il est peu vraisemblable que l'on puisse atteindre dans ce cas un équilibre de Nash en temps polynomial (v. aussi [82]). Papadimitriou concluait donc ainsi son exposé lors d'une conférence d'économie : *« if your market can compute a Nash equilibrium, I would like to meet your market. »*

[32] Je préfère algorithme *randomisé* à algorithme *aléatoire*, afin de souligner le fait que le hasard est injecté à dessein dans l'algorithme plutôt que subi.

[35] Précisément, dans [179], Kabanets et Impagliazzo démontrent que si on peut tester en temps polynomial déterministe, la nullité d'un polynôme donné sous la forme d'un circuit arithmétique (ayant pour opérateurs $+, -, \times$ d'arité 2 ; ce problème est emblématique de la classe BPP), alors ou bien il existe une fonction calculable en temps exponentiel non-déterministe qui n'est pas calculable par des circuits de taille polynomiale (non-uniforme) (c.-à-d., $NEXP \not\subseteq P/_{\text{poly}}$) ou bien le permanent d'une matrice ne peut pas être calculé par des circuits arithmétiques de tailles polynomiales (non-uniformes) (c.-à-d., perm $\notin AlgP/_{\text{poly}}$). Se reporter aux chapitres 9 et 19 à 23 du livre [18] pour plus de détails sur les implications des différentes conjectures jugées vraissemblables par la communauté scientifique dans ce domaine actuellement.

[36] A. Prochiantz (Cours n°1 au Collège de France, 2007, 17ème min 30 sec) : *"On peut faire toute sorte de modèles comme ça, qui sont évidemment très intéressants, le problème c'est de savoir si ça correspond à quelque chose de réel. Et si ça correspond pas à quelque chose de réel, savoir comment ça peut nous aider à comprendre ce qu'il se passe réellement parce que, que ce soit faux ce n'est pas grave, mais il faut que ce soit faux et que ça nous oblige à trouver des choses plutôt que ce soit faux et que ça nous empêche de trouver des choses. Ça c'est essentiellement la façon d'utiliser les modèles, ça dépend de celui qui les utilise, et aussi du dogmatisme de celui qui les propose, donc ça c'est une autre affaire"*.

1 Petits-mondes, navigabilité, émergence & réseaux pair-à-pair

Ce chapitre présente les travaux que nous avons menés sur la compréhension de l'émergence d'une propriété globale de certains réseaux d'interaction : la *navigalibité*. Ces travaux constituent le cœur de la thèse d'Emmanuelle Lebhar. Ce chapitre est composé de cinq parties : après une introduction générale et la définition du cadre de notre étude, les trois sections suivantes traitent de nos contributions à ce domaine. Les sections 3 et 5 traitant de la petit-mondisation et de l'émergence ont été réalisées en collaboration avec Philippe Duchon et Nicolas Hanusse (LaBRI).

1 Introduction

1.1 Réseaux d'interaction

La thématique des réseaux d'interaction englobe l'étude de tout graphe représentant des interactions locales entre des entités, p. ex. : un lien par participation à des réactions chimiques pour des protéines en biologie, un lien par co-publication entre co-auteurs, un lien entre deux pages web qui pointent l'une sur l'autre, un lien entre deux personnes qui se connaissent pour les réseaux sociaux, etc. Bien sûr, ces graphes peuvent être annotés par des informations supplémentaires, des arcs redondants, etc. Comme nous l'avons vu dans l'introduction, ces graphes sont depuis une grosse décennie l'objet d'études intensives rendues possibles par l'avènement de nouvelles techniques de mesure informatique. Il a ainsi été observé que ces graphes partagent souvent de propriétés originales par rapport à leur pendant aléatoire uniforme ayant le même nombre d'arêtes et de sommets (v. p. ex., [106, 32]). Typiquement, malgré un très petit nombre d'arêtes, leurs *diamètre et excentricité*[1] *moyenne sont très petits*. À titre d'exemple, le graphe des pages web comptait environ $8 \cdot 10^8$ pages en 1999, et on observait en moyenne environ $4{,}59$ hyperliens par page ; pourtant son excentricité moyenne était de l'ordre de $18{,}59$ seulement [11]. Une seconde caractéristique typique est la distribution des degrés[2] des sommets. On observe le plus souvent *une loi de puissance*,

[1] L'*excentricité* d'un sommet est sa distance au sommet qui en est le plus éloigné : $\text{excentricité}(u) = \max_v d(u, v)$.

[2] Le *degré* d'un sommet u est le nombre d'arêtes qui lui sont reliées : $\text{degré}(u) = \#\{\text{arête } e : u \in e\}$.

$\Pr\{\text{degré}(u) = d\} \propto d^{-\alpha}$, où l'exposant α varie suivant les graphes considérés (typiquement entre 2 et 3). Rappelons que ce phénomène est lié au phénomène de *scaling* (absence d'échelle).[3] Enfin, une troisième caractéristique relevée par la communauté est une *forte interconnectivité locale*.[4] Il s'agit d'une propriété typique des réseaux sociaux où "les connaissances d'une même personne se connaissent également en général". D'autres lois typiques des graphes réels ont été également observées par la suite. Par exemple, les valeurs propres de leur matrice d'adjacence suivent elles aussi une loi de puissance [106]. Il semble cependant que ce dernier fait soit directement lié à l'existence de loi de puissance sur les degrés des sommets comme le démontre [215].

Ces caractéristiques statistiques ont permis d'isoler certains comportements radicalement différents entre réseaux réels et modèles uniformes. Par exemple, dans un graphe aléatoire uniforme, une proportion minimale de personnes initialement contaminées est nécessaire pour qu'une épidémie (informatique ou biologique) contamine tous les individus. On parle de seuil épidémique. Alors que dans un graphe aléatoire dont la distribution des degrés suit une loi de puissance fixée, ce phénomène de seuil disparaît [235, 42]. Une contamination initiale infime suffit pour toucher tous les individus.

Newman le reconnaît cependant volontiers dans son état de l'art [228] en 2003 : ces trois caractéristiques statistiques (diamètre faible, loi de puissance sur les degrés, interconnectivité locale forte) ont surtout le mérite de sauter aux yeux, d'être faciles à mesurer et de démarquer franchement les réseaux "réels" des modèles probabilistes uniformes classiques, sur lesquels se concentrait la plupart des études jusque là. Ces propriétés doivent être prises en compte car elles influent sur certains comportements de ces réseaux. Mais elles restent cependant très superficielles au sens où il semble encore difficile d'en cerner à la fois les origines et les implications. De plus, leur omniprésence dans des réseaux très différents (sociaux, technologiques, biologiques,...) est à la fois intrigante et suspecte quant à leur importance réelle sur les phénomènes considérés. Certains doutes subsistent d'ailleurs quant à leur existence même. En effet l'article [8] démontre mathématiquement que la méthode de mesure du graphe d'Internet utilisé actuellement, par `traceroute`, conduit à observer des lois de puissances (artificielles) sur les degrés des sommets dans des graphes aléatoires uniformes, alors que ceux-ci suivent en réalité une loi de Poisson [102] !

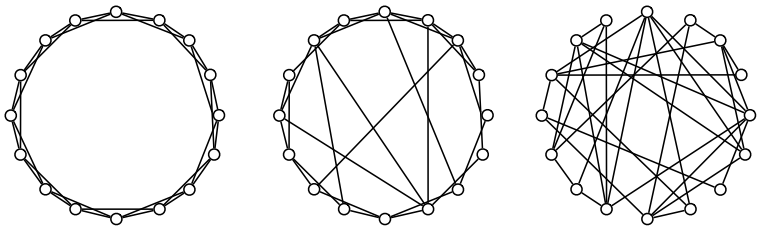
Il est donc tout à fait possible que ces propriétés ne soient qu'un bruit de fond et que d'autres propriétés moins évidentes jouent un rôle plus important dans le phénomène considéré. L'objet de ce chapitre est de démontrer qu'une approche algorithmique peut se révéler fondamentale pour révéler certains aspects des réseaux d'interaction.

1.2 Le phénomène des petits mondes

Nous nous intéresserons dans ce chapitre plus particulièrement aux graphes dits *sociaux* qui traduisent des contacts relationnels entre des individus. Précisément, notre travail trouve ses racines dans les travaux de Milgram [216]. En 1967, Milgram mena une expérience consistant à demander à 300 habitants du Nebraska de faire parvenir une lettre à un habitant de Boston dont ils ne connaissaient que le nom, le lieu d'habitation et la profession. La règle était que la lettre ne pouvait être remise qu'à une personne connue personnellement du détenteur courant (en itérant le processus jusqu'à la cible). Le résultat de l'expérience fut surprenant. Bien que

[3] v. Note 23 du chapitre 0 22.

[4] l'*interconnectivité locale* (*clustering coefficient*, en anglais) d'un sommet u [315] est la proportion d'arêtes entre ces voisins présentes dans le graphe : $\text{interconnectivité}(u) = \frac{\#\{vw : u, v \text{ et } w \text{ sont reliés entre eux}\}}{\text{degré}(u)}$.



Quantité d'aléa :	aucun ($p = 0$)	**modérée ($0 < p < 1$)**	beaucoup ($p = 1$)
Diamètre :	grand	**petit**	**petit**
Interconnectivité locale :	**forte**	**forte**	faible

Figure 1.1 – Le modèle de Watts et Strogatz [315].

seulement 20% des lettres soient arrivées à destination, elles y sont arrivées via 5,2 contacts intermédiaires en moyenne. L'expérience a été reproduite en 2003 par Dodds, Muhamad et Watts [86] sur un groupe de 60 000 individus échangeant cette fois-ci des *emails*. Elle a abouti cette fois-ci à des chaînes d'une longueur moyenne 4,1 entre individus de continents différents. On peut donc conclure que le graphe des relations sociales a certainement une très petite excentricité moyenne (5,2 pour 200 millions d'individus aux États-Unis d'Amérique en 1967). Plus intéressant, cette expérience révèle que des individus qui n'ont qu'une vue très *locale* de ce graphe (leurs contacts personnels), arrivent malgré tout à trouver de façon *décentralisée* (sans coordination globale) des chemins courts vers une cible décrite par un certain nombre de caractéristiques (lieu et profession, p. ex.).

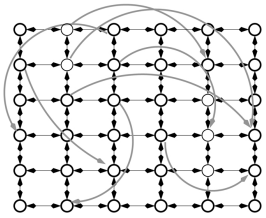
1.3 Un premier modèle des graphes petits mondes

Les premières constatations sur les graphes sociaux (petit diamètre et forte interconnectivité locale) ont conduit Watts et Strogatz [315] à proposer un premier modèle de graphe petit-monde, illustré Fig. 1.1. Partant d'un cycle de longueur n où chaque sommet est relié à ses k voisins de droite et de gauche (induisant une forte interconnectivité locale mais un grand diamètre), ils proposent de remplacer une proportion p de ces liens par des liens aléatoires uniformes. Ils constatent alors que, pour les valeurs intermédiaires de p, le graphe présente à la fois une forte interconnectivité locale et un petit diamètre. Les petits mondes seraient donc le produit de l'injection d'une quantité modérée de hasard dans une structure ordonnée (approche de type *edge of chaos*). De façon assez surprenante, ce modèle très naïf a joué un rôle historique important et reste une référence en physique des réseaux d'interaction. Ce modèle m'apparaît pourtant assez insatisfaisant au sens qu'il recherche plus à reproduire (assez artificiellement convenons-en) des paramètres observés dans la nature plutôt qu'à les expliquer.

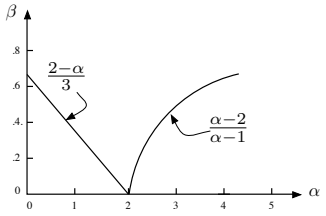
Remarquons que ce modèle est en particulier incapable d'aborder la question de la capacité des individus à trouver des chemins courts, qui est selon moi le résultat le plus frappant des travaux de Milgram. Ce dernier point est justement l'objet du travail de Kleinberg.

1.4 Navigabilité : le modèle de Kleinberg

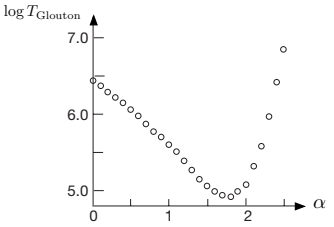
En 2000, Kleinberg [188, 190] introduit la notion de réseau *navigable*, dans lequel les individus peuvent trouver des chemins courts entre toute paire de points *en se basant uniquement sur leur vision locale* du réseau. Pour cela, il définit un modèle à base d'une grille d-dimensionnelle, réprésentant les relations de *voisinage géographique* (ou professionnel) entre



(a) Une grille bidimensionnelle augmentée suivant le modèle de Kleinberg.



(b) Valeur de l'exposant β du minorant $\Omega(n^{\beta})$ du temps d'atteinte de tout algorithme décentralisé en fonction de α (ici, $d = 2$).



(c) Valeurs expérimentales de l'espérance du temps d'atteinte par l'algorithme glouton en fonction de α pour une grille $20\,000 \times 20\,000$.

	$0 \leqslant \alpha < d$	$\alpha = d$	$0 < \alpha < 2d$	$\alpha > 2d$
Diamètre (espérance)	$\Theta(\log n)$	$\Theta(\log n)$	$\Theta(\log^{\gamma} n), \gamma > 1$	$\Theta(n^{\gamma}),\ \gamma > 0$
Algorithme glouton [188]	$\Omega(n^{\gamma}),\ \gamma > 0$	$\Theta((\log n)^2/k)$	$\Omega(n^{\gamma}),\ \gamma > 0$	
Notre algorithme [195, 196]	″	$\Theta(\log n \cdot (\frac{\log\log n}{\log k})^2)$	″	
Notre nouvel algorithme 1.4	″	$\Theta(\log n \cdot \log\log n)$	″	

(d) Espérance du diamètre d'un graphe de Kleinberg aléatoire en fonction du paramètre α [73, 206, 207] et espérances des longueurs des chemins calculés par l'algorithme glouton [188] et par nos algorithmes par explorations successives qui seront décrits à section 4[195, 196].

Figure 1.2 – Le modèle de Kleinberg (illustrations extraites de [188, 189, 187].

les individus. Cette grille est ensuite augmentée de k liens orientés aléatoires par sommet. Chacun de ces liens pointe vers un sommet aléatoire, choisi indépendamment et uniformément parmi ceux à distance d de leur origine, où d est choisie aléatoirement et indépendamment suivant une loi de puissance $\propto 1/d^{\alpha}$ (α est le paramètre du modèle). La construction est illustrée figure 1.2(a). Ces k contacts aléatoires représentent dans son modèle les personnes rencontrées au hasard de la vie.

Le fait marquant démontré par Kleinberg [188, 33] est que, même si ce graphe augmenté admet un petit diamètre pour une large plage du paramètre α (précisément, un diamètre polylogarihmique en n pour $\alpha \in [0, 2d)$, v. [73, 206, 207]), *ce graphe n'est navigable que pour la valeur* $\alpha = d$ (v. figure 1.2(b) et 1.2(c)) :

- *pour tout* $\alpha \neq d$ *: tout algorithme décentralisé* calculera *nécessairement* des chemins de longueur moyenne polynomiale, $\Omega(n^{\beta(\alpha)})$ pour une certaine fonction $\beta(\alpha) > 0$;
- *tandis que pour* $\alpha = d$ *:* il existe un algorithme très simple, l'algorithme *glouton* qui choisit systématiquement de suivre le lien (local ou aléatoire) qui le rapproche le plus de la cible (au sens géographique, *i.e.*, de la distance dans la grille non-augmentée), qui calcule des chemins très courts, de longueur $\Theta((\log n)^2/k)$ en espérance.

Kleinberg en déduit en particulier que *la famille de graphes de Watts et Strogatz (Fig. 1.1) n'est pas navigable*. En effet, comme les liens sont reroutés de façon uniforme au hasard, ces graphes correspondent tous à la valeur $\alpha = 0$ ($< d$) du paramètre ; et donc, bien que leurs diamètres soient logarithmiques, tout algorithme ayant une vision purement locale calculera systèmatiquement des chemins de longueur polynomiale. Le modèle de Watts et Strogatz ne

peut donc prétendre à la qualification de “petits-mondes” dans le sens donné par l'expérience de Milgram.

L'approche algorithmique de Kleinberg apporte donc un éclairage nouveau sur ce phénomène en définissant précisément la *fonctionnelle* correspondant au phénomène que l'on désire comprendre : la capacité à trouver des chemins courts à partir d'informations purement locales, capacité que nous définirons formellement à la section 3.

Petits mondes de Kleinberg et réseaux pair-à-pair. J'avais assisté à la conférence de Kleinberg introduisant avec humour et persuasion son modèle à STOC en mai 2000 [188] et comme toute la salle, nous avions trouvé ce résultat étonnant, intéressant, mais nous n'avions pas la moindre idée de ce que nous pourrions en faire. Il s'est avéré que quelques années après, les premiers résultats dérivés parurent avec un impact extraordinaire en algorithmique distribuée et tout particulièrement sur les réseaux pair-à-pair. En effet, Kleinberg propose avec son modèle non seulement une explication convaincante d'un phénomène social, mais aussi un cadre extrêmement simple, économique et efficace pour le routage dans les réseaux pair-à-pair. Les réseaux communautaires de type pair-à-pair sont par nature très volatiles et les nœuds apparaissent ou disparaissent de façon erratique. Maintenir une vision globale du réseau aurait alors un coût prohibitif en terme d'efficacité. Le but est alors d'adopter un contrôle totalement décentralisé permettant l'apparition et la disparition spontanées des nœuds, tout en tentant de garantir un accès rapide à chacun des nœud du réseau (v.p. ex. Chord [297] ou Tapestry [333]). Le modèle de Kleinberg se prête parfaitement à ce cadre. Les allers et retours entre applications aux réseaux informatiques et modélisation sociale ont permis une irrigation réciproque de ces deux domaines qui a produit et continue de produire des résultats très riches. Il en a résulté simultanément un éclairage original sur une thématique technique (l'optimisation des réseaux virtuel) et des techniques algorithmiques pour l'étude de phénomènes sociaux [281] (v. p. ex. l'article tout récent [60] dont nous reparlerons à la conclusion de ce chapitre).

1.5 Notre contribution

Avec Emmanuelle Lebhar puis Nicolas Hanusse et Philippe Duchon, nous nous sommes intéressés à trois aspects du phénomène petit-monde : les aspects dynamique, structurel et l'émergence.

L'aspect structurel. Kleinberg proposait dans son modèle d'augmenter une grille suivant une loi harmonique pour la transformer en “petit monde”, c.-à-d. en un graphe dans lequel on peut trouver de façon décentralisée des chemins très courts (polylogarithmique en le nombre d'individus). Avec Nicolas Hanusse, Emmanuelle Lebhar et Philippe Duchon, nous avons proposé de généraliser cette notion en introduisant la notion de *petit-mondisation*. Il s'agit de processus qui ajoutent un certain nombre de liens aléatoires à un graphe de sorte que ces liens, connus de leurs origines uniquement, permettent, ensemble, de calculer des chemins très courts (polylogarithmiques) en se basant uniquement sur la connaissance du graphe initial et des liens rencontrés au fur et à mesure de la progression. Nous nous sommes alors posé trois types de questions : la petit-mondisation est-elle toujours possible ? quelles propriétés structurelles doit posséder un graphe pour qu'elle soit possible ? comment procéder pour l'obtenir quand elle existe ?

Nous avons exploré l'importance de la métrique induite par un graphe sur ses sommets dans l'existence d'un tel schéma d'augmentation. Deux paramètres nous ont semblé importants : sa croissance (au sens de la croissance du nombre de sommets situés à distance $\leqslant r$

d'un sommet donné) et sa capacité à être plongé dans un espace euclidien. Dans [88, 89] (v. section 3), nous avons démontré que tout graphe à *croissance bornée* (définition 1.9) est petit-mondisable, généralisant ainsi le modèle de Kleinberg à une métrique arbitraire. Nous avons par la suite également proposé un schéma à base de plongement dans des espaces euclidiens (v. section 3.2, [194, Chap. 5.6] et [197]).

Un corollaire intéressant de nos travaux est qu'ils démontrent que la distribution des degrés des sommets du graphe n'a quasiment aucune influence sur la navigabilité d'un réseau (un degré moyen constant suffit). Ainsi, contrairement aux phénomènes de seuils épidémiques, la navigabilité se révèle être indépendante de l'existence de loi de puissance sur les degrés.

L'aspect dynamique. Nous nous sommes intéressés à l'algorithme de routage considéré par Kleinberg. Il nous a semblé qu'il ne correspondait pas nécessairement à la réalité et que d'autres comportements devaient être envisagés. Typiquement, il nous a semblé irréaliste de considérer que chacun se dirige systématiquement vers le contact qui est *à sa connaissance* le plus proche de la cible, *sans se renseigner* auparavant auprès de son entourage. Typiquement, dans le cas du web, nous procédons le plus souvent par approximations successives lorsque l'on recherche une information. Plutôt que de suivre systématiquement le premier lien qui nous rapproche un peu de la page que l'on recherche, nous partons d'une page qui nous apparaît pertinente, puis nous cliquons sur un des liens qu'elle contient, regardons s'il est vraiment plus pertinent, et dans le doute, on revient en arrière, pour en cliquer d'autres, jusqu'à en trouver un qui nous fasse vraiment avancer vers ce que l'on cherche. Ce genre de stratégie évite de se perdre dans des culs-de-sac.

Nous avons donc proposé un algorithme [195, 196] (v. section 4) qui s'autorise une phase d'exploration avant de choisir de suivre le lien qui paraît être le meilleur. Nous avons démontré que cet algorithme produit de façon décentralisée des chemins considérablement plus courts, de longueur quasi-optimale $O(\log n \cdot (\frac{\log \log n}{\log k})^2)$ au lieu des $O(\log^2 n / k)$ pour le glouton. Cette propriété est particulièrement intéressante dans le cas des transferts de gros fichiers sur un réseau pair-à-pair, où la longueur du chemin à traverser est déterminante pour les temps de transfert et la charge induite sur les liens du réseau.

En écrivant ce manuscrit, je me suis aperçu que l'on pouvait améliorer considérablement la présentation de cet algorithme en adoptant un point vue purement géométrique. Cette nouvelle présentation m'a ensuite permis d'en améliorer les performances et la généralité. Nous présenterons donc ici cette nouvelle version de notre algorithme.[5]

Nous nous sommes ensuite intéressés à l'impact de l'algorithme de routage sur la charge des liens du réseau et cela pour deux raisons. Tout d'abord, nous souhaitions évaluer les risques de saturation induits par les différents algorithmes. Mais surtout car les calculs menés par Lebhar dans sa thèse [194, Chap. 3] ont démontré que ce paramètre peut être utilisé pour déterminer quel algorithme est le mieux adapté pour modéliser le comportement des individus dans un graphe réel. Cette étude ouvre donc la possibilité de valider les modèles comportementaux algorithmiques proposés.

L'émergence. L'inconvénient majeur du modèle de Kleinberg est son absence d'explication sur les raisons de l'apparition du phénomène des petits mondes.

On peut imaginer a priori une explication très simple. La taille d'une société est limitée par la performance de ses communications internes. Au-delà d'une certaine taille, des incohérences apparaissent et la société se disloque. La sélection naturelle fait que les sociétés ayant

[5] Depuis la rédaction de ce document, nous avons démontré avec George Giakkoupis que cette approche est en fait optimale [134].

les meilleurs réseaux de communication se développent mieux que d'autres. Un tel exemple a été observé tout récemment sur les protocoles pair-à-pair. Initialement, le protocole Gnutella 0.4 [136] procédait par inondation systèmatique du réseau pour le traitement de chaque requête. Ce procédé limitait de fait la taille des réseaux constitués à un petit millier de nœuds. Au-delà, les performances se dégradaient tellement que les utilisateurs créaient spontanément d'autres réseaux séparés.[6]

Cependant, si cette explication est satisfaisante pour les systèmes conçus à dessein (une entreprise, etc.), elle est très largement insatisfaisante pour les relations amicales ou sociales : on ne choisit pas (en général) ses co-auteurs pour minimiser son Erdös-number. Nous nous sommes donc posés la question de savoir si le modèle de Kleinberg pouvait *émerger* naturellement dans un système social, où l'on compte seulement un millier de contacts par individu en moyenne pour un total de plusieurs milliards d'individus.

Avec Nicolas Hanusse, Emmanuelle Lebhar et Philippe Duchon, nous nous sommes précisément intéressés à la possibilité de construire le modèle de Kleinberg rapidement sous ces contraintes. Dans [90] (v. section 5), nous avons démontré qu'il était possible de transformer tout graphe à croissance bornée en un petit-monde, par l'ajout d'un unique lien aléatoire par nœud, déterminé de façon totalement décentralisée, en temps polylogarithmique (en fonction du nombre de sommets) et ne consommant qu'une mémoire polylogarithmique en chaque nœud.

Concrètement, nous avons obtenu un protocole réseau totalement distribué qui, en un temps polylogarithmique et en ne consommant qu'un nombre polylogarithmique de bits de mémoire localement, ajoute *une seule* entrée à chaque table de routage locale qui permet ensuite de découvrir localement des routes de longueur polylogarithmique entre toutes les paires de sommets du graphe, c.-à-d. sans modifier en rien les algorithmes de routage présents dans les routeurs.

2 Définitions et notations

Soient n et d deux entiers strictement positifs.

Définition 1.1 (Tore et distance ℓ_1). *On appelle* tore d-dimensionnel de taille n, *le graphe non-orienté* $\mathcal{T}_n^d = (V_n^d, E_n^d)$ *de sommets* $V_n^d = (\mathbb{Z}/n\mathbb{Z})^d$, *où* $uv \in E_n^d$ *ssi il existe* $i \in \{1, \ldots, d\}$ *tel que* $u_i = v_i \pm 1 \bmod n$ *et* $u_j = v_j$ *pour* $j \neq i$.

Pour u *et* v *dans* V_n^d, *on note* $||u - v||$ *la distance entre* u *et* v *dans le tore* $\mathcal{T}_n^d$, *appelée également la* distance ℓ_1 *entre* u *et* v.

Définition 1.2 (Modèle de Kleinberg). *Étant donnés* $\alpha \in \mathbb{R}_+^*$ *et un entier* $k > 0$, *on appelle* modèle de Kleinberg d-dimensionnel de taille n à k longs liens et de paramètre α, *le graphe aléatoire* $\mathcal{K}_{n,k}^{d,\alpha}$ *obtenu en ajoutant à chaque sommet* u *du tore* $\mathcal{T}_n^d$, k *liens orientés aléatoires. Chacun de ces liens pointe vers un sommet aléatoire* v *i.i.d.*[7] *choisi suivant la loi :*

$$\Pr\{u \to v\} = \frac{1}{Z_n^{d,\alpha}} \cdot \frac{1}{||u - v||^\alpha} \text{ pour } v \neq u,$$

[6] Ce procédé par inondation, corrigé dans la version 0.6 du protocole, faisait d'ailleurs que Gnutella occupait près de 80% de la bande passante d'Internet sans que son usage représente 80% des données effectivement transférées comme tentaient de laisser croire certain lobbies alarmistes.

[7] L'abréviation *i.i.d.* signifie indépendants et identiquement distribués.

où $Z_n^{d,\alpha}$ *est la fonction de partition*[8] *associée* $Z_n^{d,\alpha} = \sum_{v \in V_n^d - \{0^d\}} \frac{1}{||v||^\alpha}$.

Pour tout sommet u *de* $\mathcal{K}_{n,k}^{d,\alpha}$, *les voisins de* u *dans le tore sous-jacent sont appelés les* contacts locaux *de* u, *et ses* k *voisins aléatoires sont appelés ses* contacts distants.[9] *Les arêtes entre contacts locaux sont appelées* liens courts *et les autres,* liens longs.

Lorsque $\alpha = d$, *on parle simplement du modèle de Kleinberg* d*-dimensionnel à* k *longs liens noté* $\mathcal{K}_{n,k}^{d}$ *et on supprime l'indice* α *des notations.*

Dans la section 3, nous utiliserons la notion suivante de graphe augmenté qui généralise le modèle de Kleinberg.

Définition 1.3 (Graphe augmenté). *Étant donné un ensemble aléatoire* F *d'arêtes sur* V *(éventuellement orientées),* le graphe augmenté $G = (V, E, F)$ *est le graphe aléatoire obtenu à partir d'un graphe* $H = (V, E)$, *en lui ajoutant l'ensemble* F *d'arêtes supplémentaires. On appelle* distance sous-jacente *de* u *à* v *dans* G, *notée* $\delta_H(u, v)$, *la distance de* u *à* v *dans* H, *excluant les arêtes aléatoires ajoutées.*

Le graphe sous-jacent représente les connaissances communes à tous, p. ex. les positions géographiques des nœuds ou leurs professions. Les arêtes supplémentaires F représentent les contacts distants (p. ex., des nœuds rencontrés au hasard de la vie), connus uniquement de l'origine des liens. On suppose que la relation de connaissance n'est pas nécessairement symétrique : p. ex., Erdös est connu au-delà du cercle de ses propres connaissances.

Définition 1.4 (Algorithme de routage). *Un* algorithme de routage *est un algorithme qui étant donnés un graphe* $G = (V, E)$, *une* source $s \in V$ *et une* cible $t \in V$ *(appelée aussi* destination*), calcule un chemin de* s *à* t *dans* G.

La définition suivante propose un cadre formel pour l'expérience de Milgram en définissant "router un message dans un graphe dont on découvre les contacts distants au fur et à mesure que le message à transmettre progresse vers la cible".

Définition 1.5 (Algorithme décentralisé, [188]). *Un algorithme de routage dans un graphe augmenté* $G = (V, E, F)$ *est* décentralisé *ssi il a pour seules connaissances initiales le graphe sous-jacent* $H = (V, E)$ *et les positions de la source et de la destination dans* H *et ne peut prendre connaissance que des voisins distants des nœuds déjà explorés.*

On appelle alors latence, *la valeur maximale de l'espérance du nombre de nœuds interrogés sur leurs contacts pour calculer un chemin entre deux sommets du graphe augmenté* G.

Un exemple d'algorithme décentralisé est l'*algorithme glouton* proposé par Kleinberg [188] (algorithme 1.1) : partant de s, tant que l'on n'est pas arrivé à la destination, prolonger le chemin courant par le contact local ou distant du sommet courant le plus proche de la cible au sens du graphe sous-jacent connu $H = (V, E)$.

Le résultat principal de Kleinberg [188], généralisé à $d \neq 2$ par [33], est le suivant :

Théorème 1.6 (Kleinberg [188, 187, 33]). *Dans le graphe augmenté* $\mathcal{K}_{n,k}^{d,\alpha}$,

(a) *si* $\alpha \neq d$, *il existe une constante* $\gamma > 0$ *telle que la latence de* tout *algorithme de routage* décentralisé *est au moins* $\Omega(n^\gamma)$;

[8] Le terme "fonction de partition" désigne la constante de normalisation d'une loi.

[9] Remarquons que les contacts locaux et distants d'un sommet u ne sont pas nécessairement disjoints.

Algorithme 1.1 Algorithme glouton de Kleinberg pour le graphe $\mathcal{K}_{n,k}^{d}$ [188]

entrée(s) : une source s et une cible t.
initialisation : $x := s$.
tant que $x \neq t$ **faire**
Passer le message au contact local ou distant y de x qui minimise $||y - t||$ (on choisit un contact arbitraire en cas d'égalité). Poser $x := y$.

(b) *si $\alpha = d$, l'algorithme décentralisé 1.1 calcule un chemin de longueur $\Theta(\log n \log \delta / k)$ en espérance entre toute paire de sommets à distance δ l'un de l'autre dans le tore sous-jacent.*

La preuve étudie trois régimes distincts. *Pour $\alpha < d$, les longs liens sont trop aléatoires*. Il est alors nécessaire de tester un nombre polynomial de sommets avant de réussir à trouver un long contact qui pointe à une distance $\leqslant n^{\gamma}$ de la cible, pour un certain $\gamma > 0$ bien choisi. *Pour $\alpha > d$, les liens sont trop courts* avec une probabilité polynomiale. Il est alors improbable de trouver en temps $\leqslant n^{\gamma}$, un lien qui nous éloigne suffisamment de la source, pour un certain $\gamma > 0$ bien choisi. *Lorsque $\alpha = d$, il suffit alors d'explorer $\Theta(\log n / k)$ nœuds* pour trouver un long lien qui pointe *vers un nœud à distance moitié de la cible*, avec probabilité constante. Ainsi l'algorithme glouton divise en moyenne la distance par deux tous les $O(\log n\, k)$ nœuds. Il calcule donc des chemins de longueur $\Theta((\log n \log \delta)/k)$ en espérance entre toute paire de sommets à distance δ dans le tore sous-jacent.

Le fait frappant révélé Kleinberg est que même si le diamètre du graphe augmenté est logarithmique, il peut être impossible de trouver ces chemins courts de façon décentralisée. En effet, il a été démontré ensuite par [73, 206, 207] (Fig. 1.2) que le diamètre du graphe augmenté de Kleinberg est logarithmique pour $0 \leqslant \alpha \leqslant d$, polylogarithmique pour $d < \alpha < 2d$ et devient polynomial seulement pour $\alpha > 2d$ (le cas $\alpha = 2d$ est encore ouvert). Pourtant, aucun algorithme décentralisé ne peut trouver des chemins de longueur polylogarithmique dès que $\alpha \neq d$.

Ce résultat a naturellement suscité un très grand intérêt dans la communauté de l'algorithmique distribuée car il touche au cœur de sa problématique. Il révèle l'impact d'une connaissance extrêmement partielle de la structure globale d'un graphe sur les performances algorithmiques.

3 Petit-mondisation

Le résultat de Kleinberg démontre qu'il est possible de transformer une grille de dimension d en un petit monde, par l'ajout d'un unique lien supplémentaire par nœud, tiré suivant une loi bien choisie. Précisément, son modèle augmente une quelconque grille en un graphe où l'algorithme glouton décentralisé engendre des chemins de longueur $O(\log^2 n)$ en espérance entre chaque paire de nœuds. Le fait que la longueur du chemin calculée soit indépendante de la dimension de la grille et que la loi qui permet de "petit-mondiser" cette grille soit liée à la dimension de la grille (la relation $\alpha = d$), suggère qu'il existe un schéma plus général recouvrant une plus grande famille de graphes.

Avec Philippe Duchon, Nicolas Hanusse et Emmanuelle Lebhar, nous nous sommes intéressés à généraliser ce résultat à des familles graphes arbitraires. Nous avons démontré que si le cardinal des boules du graphe ne croît pas trop vite, c.-à-d. s'il a une croissance à peine

plus rapide que polynomiale en fonction du rayon, alors nous pouvons construire un schéma très simple de petit-mondisation, basé sur la topologie locale du graphe.

Conformément à la définition 1.3 page 32, nous cherchons à augmenter un graphe de base $H = (V, E)$ avec un ensemble aléatoire d'arcs F de manière à ce qu'un algorithme décentralisé, connaissant initialement uniquement la métrique de H, puisse engendrer des chemins très courts entre toute paire de nœuds du graphe augmenté avec forte probabilité.[10] Nous supposerons que $\delta_H(u, v)$ est donnée par un oracle, c.-à-d. qu'elle peut être calculée à partir des étiquettes de u et v comme c'est le cas pour la grille d-dimensionnelle (v. p. ex. [4, 5, 133, 303] pour des constructions classiques). Nous ne considérons que des schémas d'augmentation qui ne rajoutent qu'un unique arc par nœud et démontrons que cela suffit pour une famille très large de graphes.

Définition 1.7 (version avec forte probabilité de [88, 89]). *Nous dirons qu'une famille infinie $\{G_n = (V_n, E_n, F_n)\}$ de graphes finis augmentés aléatoirement à partir d'une famille de graphes de base $H_n = (V_n, E_n)$, où n désigne abusivement le nombre de sommets de H_n, est une* famille de petits mondes navigables *s'il existe un polynôme p et un algorithme décentralisé, utilisant uniquement la métrique sous-jacente δ_{H_n} de H_n, qui engendre avec probabilité $1 - O(\frac{1}{n})$ des chemins de longueur au plus $p(\log n)$ entre toutes les paires de nœuds dans G_n.*

Définition 1.8 (Schéma de petit-mondisation). *On appelle* schéma de petit-mondisation *d'une famille de graphes finis $\{H_n\}$, tout algorithme qui associe à chaque sommet u de H_n un sommet aléatoire $L_{n,u}$ tel que la famille de graphes augmentés aléatoires $G_n = (V_n, E_n, F_n)$ forme un petit-monde navigable, où F_n est l'ensemble des arcs $u \to L_{n,u}$ pour $u \in V_n$.*

Lorsque le contexte le permettra, nous abandonnerons l'indice n dans les notations et utiliserons n pour désigner le nombre de sommets du graphe.

3.1 Petit-mondisation des graphes à croissance bornée

Nous nous intéressons dans un premier temps aux graphes à croissance bornée, c.-à-d. ceux dont les tailles de boules restent dans des ratios comparables aux ratios de leurs rayons. Nous noterons $B_{H_n,u}(r) = \{v : \delta_{H_n}(u, v) \leqslant r\}$ la boule de rayon r centrée sur le nœud u dans le graphe H_n, et $b_{H_n,u}(r) = \#B_{H_n,u}(r)$ son cardinal. Lorsque le contexte le permet, nous supprimerons H_n de l'indice. Notons que l'on ne considérera jamais les boules de G_n (dont les algorithmes décentralisés ignorent tout) mais seulement les boules du graphe sous-jacent H_n.

La définition suivante présente à la fois une généralisation et une simplification de la notion de graphes à croissance ρ-modérée introduite dans [88, 89], au sens que les graphes à croissance $O(1)$-modérée sont à croissance polylog(r)-modérée.

Définition 1.9 (Graphe à croissance $\alpha(r)$-bornée). *Nous dirons qu'un graphe H est* à croissance $\alpha(r)$-bornée *si pour tout nœud u et pour tout $r \geqslant 1$, $b_u(r) \leqslant \alpha(r)\, b_u(r/2)$.*

Cette définition est très utilisée dans la littérature car il s'est avéré que de nombreux graphes réels vérifiaient cette propriété, p. ex. lorsqu'ils sont contraints géographiquement comme pour

[10] Nous dirons qu'un événement se produit avec *forte probabilité* si sa probabilité est de l'ordre de $1 - O(\frac{1}{n})$.

le graphe d'Internet. On peut citer p. ex. les travaux de [155, 293] qui utilisent ces modèles et ceux de [122] pour une confrontation de ces modèles à la réalité mesurée d'Internet.

L'algorithme 1.2 présente notre schéma de petit-mondisation pour les graphes à croissance polylog(r)-bornée.

Algorithme 1.2 Petit-mondisation des graphes à croissance $O(\log^\rho r)$-bornée [88, 89]

entrée(s) : Un graphe fini $H = (V, E)$

pour tout $u \in V$ **faire**

Tirer aléatoirement et indépendamment le contact distant L_u de u suivant la loi :

$$\Pr\{L_u = v\} = \frac{1}{Z_u} \cdot \frac{1}{b_u(r)}$$

où $r = \delta_H(u, v)$ et Z_u est la fonction de partition associée :

$$Z_u = \sum_{v \in V : v \neq u} \frac{1}{b_u(\delta_H(u,v))} = \sum_{r \geqslant 1} \frac{b_u(r) - b_u(r-1)}{b_u(r)}.$$

Théorème 1.10 ([88, 89]). *L'algorithme 1.2 est un schéma de petit-mondisation pour toute famille de graphes $\{H_n = (V_n, E_n)\}$ à croissance $O(\log^\rho r)$-bornée.*

Plus précisément, si n désigne le nombre de sommets de H_n, avec probabilité $1 - O(\frac{1}{n})$, l'algorithme glouton de Kleinberg calcule des chemins de longueur $O(\log^{2+4\rho} n)$ entre toutes les paires de nœuds dans le graphe augmenté $G_n = (V_n, E_n, F_n)$, où F_n désigne l'ensemble $\{u \to L_u : u \in V_n\}$.

Démonstration. Considérons H_n à croissance $\alpha(r)$-bornée avec $\alpha(r) = O(\log^\rho r)$, et désignons par n, le nombre de sommets du graphe. Évaluons tout d'abord la fonction de partition Z_u en regroupant les termes de la somme par paquets exponentiellement croissants.[11] Par croissance de $r \mapsto b_u(r)$ et sommation téléscopique, nous obtenons que :

$$Z_u = \sum_{i=0}^{\lfloor \log_2 n \rfloor} \sum_{2^i \leqslant r < 2^{i+1}} \frac{b_u(r) - b_u(r-1)}{b_u(r)} \leqslant \sum_{i=0}^{\lfloor \log_2 n \rfloor} \frac{1}{b_u(2^i)} (b_u(2^{i+1} - 1) - b_u(2^i - 1)).$$

Ainsi,

$$Z_u \leqslant \sum_{i=0}^{\lfloor \log_2 n \rfloor} \frac{b_u(2^{i+1})}{b_u(2^i)} \leqslant \sum_{i=0}^{\lfloor \log_2 n \rfloor} \alpha(2^{i+1}) = \sum_{i=0}^{\lfloor \log_2 n \rfloor} O(i^\rho) = O(\log^{\rho+1} n).$$

Considérons maintenant le routage glouton de Kleinberg dans le graphe augmenté aléatoirement G_n. Supposons que l'on ait atteint un nœud u à distance r de la cible t dans H_n, nous allons minorer la probabilité que le contact distant $L_{n,u}$ de u soit à distance $\leqslant r/2$ de t

[11] Dans [88, 89], nous n'avions pas procédé ainsi et avions été obligés d'introduire une définition de croissance *modérée* plus contraignante pour obtenir une majoration uniforme de Z_u.

dans H_n. Par construction de G_n,

$$\begin{aligned}\Pr\{L_{n,u} \in B_{H_n,t}(r/2)\} &= \frac{1}{Z_u} \sum_{v \in B_{H_n,t}(r/2)} \frac{1}{b_{H_n,u}(\delta_{H_n}(u,v))} \\ &\geqslant \frac{1}{Z_u} \frac{b_{H_n,t}(r/2)}{b_{H_n,u}(3r/2)} \geqslant \frac{1}{Z_u} \frac{b_{H_n,t}(r/2)}{b_{H_n,t}(5r/2)},\end{aligned}$$

car nous avons $B_{H_n,t}(r/2) \subseteq B_{H_n,u}(3r/2) \subseteq B_{H_n,t}(5r/2)$ dans H_n (v. fig. 1.3). Or,

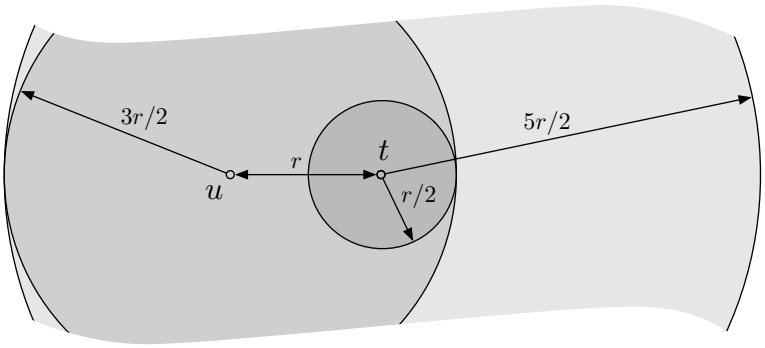


Figure 1.3 – Inclusion des boules dans la preuve du théorème 1.10.

comme H_n est à croissance $O(\log^\rho r)$-bornée, le ratio des tailles des boules de rayons $r/2$ et $5r/2 \leqslant 2^3\, r/2$ centrées sur t est borné par $O(\log^{3\rho} r)$ et :

$$\Pr\{L_{n,u} \in B_{H_n,t}(r/2)\} = \Omega\left(\frac{1}{\log^{\rho+1} n \times \log^{3\rho} r}\right) = \Omega\left(\frac{1}{\log^{1+4\rho} n}\right).$$

L'algorithme glouton suit toujours le lien local ou distant qui rapproche le plus de la cible. À chaque pas, la distance du point courant à la cible est donc divisée par 2 avec probabilité au moins $p_n = \Omega(1/\log^{1+4\rho} n)$, indépendante des positions relatives de u et t. Considérons donc une suite de variables aléatoires i.i.d. $(X_i)_{i>0}$ à valeurs dans $\{0,1\}$ telles que $\Pr\{X_i = 1\} = p_n$. Il s'en suit que le nombre de pas effectués par l'algorithme glouton est dominé stochastiquement par la variable aléatoire :[12]

$$T = \min\left\{t : \sum_{i=1}^{t} X_i \geqslant \log_2 n\right\}$$

[12] On dit qu'une variable aléatoire X à valeurs réelles est *dominée stochastiquement* par une variable aléatoire Y si pour tout $x \in \mathbb{R}$, $\Pr\{X \geqslant x\} \leqslant \Pr\{Y \geqslant x\}$. Le procédé habituel pour prouver la domination stochastique est de coupler les deux variables aléatoires sur le même espace probabilisé Ω de telle sorte que pour tout événement $\omega \in \Omega$, $X(\omega) \leqslant Y(\omega)$. Dans ce mémoire, nous omettrons la plupart du temps de détailler le couplage sous-jacent car sa construction ne présente en général aucune difficulté.

Or l'inégalité de Chernoff [225, section 4.1 page 67] donne que :

$$\begin{aligned}\Pr\{T > 9\log_2 n/p_n\} &= \Pr\left\{\sum_{i=1}^{9\log_2 n/p_n} X_i < \log_2 n\right\}\\ &\leqslant \Pr\left\{\sum_{i=1}^{9\log_2 n/p_n} X_i < \frac{1}{2}\cdot \underbrace{\frac{9\log_2 n}{p_n}\cdot p_n}_{\mathbb{E}\left[\sum_i X_i\right]}\right\}\\ &\leqslant e^{-\frac{1}{4}\times 9\log_2 n} \leqslant \frac{1}{n^3}.\end{aligned}$$

Ainsi pour tout couple (s,t), l'algorithme glouton calcule un chemin de s à t dans G_n de longueur $O(\log n\,/p_n) = O(\log^{2+4\rho} n)$ avec probabilité $1-1/n^3$. En prenant la probabilité de l'union des échecs sur les $n(n-1)$ couples, on en déduit qu'avec probabilité $1-\frac{1}{n}$, l'algorithme glouton calcule donc un chemin de cette longueur entre tous les couples de nœuds. La famille de graphes $\{G_n\}$ forme donc bien un petit monde navigable et ce quelque soit le diamètre initial des graphes H_n. ■

Remarquons que pour les grilles d-dimensionnelles pour lesquelles $\rho = 0$, nous retrouvons le résultat de Kleinberg [188, 33] (légèrement amélioré puisque nous démontrons que la borne $O(\log^2 n)$ sur la longueur du chemin est vraie simultanément pour tous les couples de nœuds avec probabilité $1-\frac{1}{n}$ et pas seulement en espérance).

Nous verrons par la suite à la section 4 que l'on peut bâtir pour cette famille de petits mondes des algorithmes très simples calculant des chemins quasi-optimaux en espérance entre toutes paires de nœuds.

Croissance $\alpha(r)$-bornée et dimension doublante. Slivkins [293] a proposé indépendamment un schéma de petit-mondisation pour la classe plus générale des graphes de dimension doublante $O(\log\log n)$.[13] Par définition, les graphes à croissance $O(\log^\rho r)$-bornée sont exactement la famille des graphes de dimension doublante *isotrope* $\alpha_{\text{isotrope}} = O(\rho\log\log n)$, au sens où toute boule B de rayon r est, avec probabilité $1-1/n$, couverte par un sous-ensemble aléatoire de $2^{O(\rho\log\log r)}$ boules de rayon $r/2$ centrées sur des nœuds choisis aléatoirement et uniformément dans B. Cette isotropie nous a permis d'obtenir explicitement la distribution de liens, contrairement au résultat de Slivkins [293] qui ne propose qu'un schéma implicite de petit-mondisation. L'isotropie sera par la suite fondamentale pour définir notre schéma de petit-mondisation décentralisé, efficace et économe à la section 5. L'existence d'une construction distribuée et économe est un premier pas vers la validation du modèle de Kleinberg en tant que modèle naturel des petits mondes observés dans la nature.

La question de la distribution des degrés et de l'interconnectivité locale. Remarquons que dans notre schéma de petit-mondisation, il suffit d'ajouter indépendamment à chaque sommet un nombre de contacts distants constant et strictement positif *en espérance*. Nous pouvons donc engendrer n'importe quelle distribution de degrés dans le graphe augmenté et même rajouter des arêtes pour obtenir l'interconnectivité locale que l'on souhaite. Notre résultat démontre donc que *la navigabilité d'un réseau est une propriété orthogonale à l'existence de loi de puissance sur les degrés ou d'une forte interconnectivité locale*. Ceci

[13] La *dimension doublante* $\alpha(H)$ d'un graphe H est la borne inférieure des réels $\alpha > 0$ tels que l'on puisse recouvrir toute boule de rayon r dans H par au plus 2^α boules de rayon $r/2$.

contredit de nombreuses idées reçues sur la question, en particulier celles véhiculées par le parallèle malencontreux avec les phénomènes d'épidémie, où ces paramètres statistiques semblent jouer un rôle.

Il est intéressant de noter que le phénomène des petits mondes révèle donc sa vraie nature uniquement au travers de l'analyse algorithmique. En particulier, c'est par l'analyse algorithmique que Kleinberg [187] a pu démontrer p. ex. que le modèle de Watts et Strogatz [315] ne peut en aucun cas rendre compte de la navigabilité observée expérimentalement dans les réseaux sociaux par Milgram [216], alors qu'il reproduit assez fidèlement les valeurs des paramètres statistiques couramment observés dans ces réseaux. L'analyse algorithmique a donc clairement un rôle important à jouer dans la compréhension des phénomènes dits complexes observés dans la nature.

3.2 Petit-mondisation par plongement

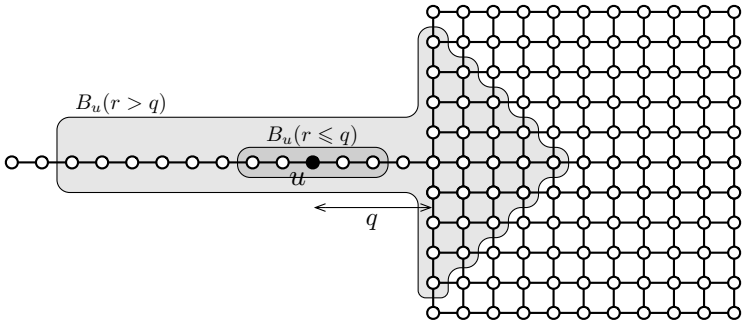


Figure 1.4 – La croissance immodérée des boules dans la tapette à mouche.

Le schéma de petit-mondisation précédent utilise le fait que la croissance du cardinal des boules est faible quand on en double le rayon. Certains graphes généralisant naturellement les grilles comme la "tapette à mouche" (fig. 1.4) ne sont pas de croissance $O(\log^\rho r)$-bornée. Les nœuds sur le manche de la tapette à mouche, situés à distance q de la jonction avec la tapette, ont des boules de taille $2r+1$ pour un rayon $r \leqslant q$ puis $\Theta(r^2)$ pour $r \geqslant (1+\varepsilon)q$. Le ratio des boules de rayons r et $2r$ est donc de l'ordre de $\Theta(r)$ autour de $r = q$. Pourtant ces graphes sont naturellement petit-mondisables en traitant le manche et le plateau de la tapette séparément par le schéma précédent. Nous proposons ici un schéma de petit-mondisation simple exploitant une autre caractéristique de ce type de graphes : le fait qu'ils soient plongeables avec peu de distorsion dans un espace ℓ_p de faible dimension.

Définition 1.11 (Plongement). *Nous dirons que $\sigma : V \to \mathbb{R}^d$ est un* plongement ℓ_p de distorsion $\gamma \geqslant 1$ *d'un graphe $H = (V, E)$ dans $\mathbb{R}^d$ si pour tout $u, v \in V$:*

$$\delta_H(u,v) \leqslant ||\sigma(u) - \sigma(v)||_p \leqslant \gamma \cdot \delta_H(u,v)$$

où $||x||_p = \left(\sum_{i=1}^d |x_i|^p\right)^{1/p}$ désigne la norme ℓ_p de $x \in \mathbb{R}^d$.

La tapette à mouche admet p. ex. un plongement sans distorsion (*i.e.*, $\gamma = 1$) dans $(\mathbb{R}^2, \ell_1)$. Depuis quelques années et l'exposé de Indyk à FOCS en 2001 [171] en particulier, les plongements de graphes (et de métriques en général) font l'objet d'une attention toute particulière en informatique, car l'obtention de plongement de faible distorsion dans

des espaces de petite dimension permet souvent de simplifier de nombreux problèmes algorithmiques. Une des premières applications algorithmiques de ces méthodes est due à Linial, London et Rabinovich [201] et Aumann et Rabani [20] qui proposèrent indépendamment une $O(\log n)$-approximation polynomiale pour le problème de la coupe la moins dense. Cette approximation est basée sur le résultat séminal de Bourgain [53] démontrant que tout graphe peut être plongé avec une distorsion $O(\log n)$ dans un espace ℓ_1 de dimension $O(\log^2 n)$ (v. [309, Chap. 21]). Depuis, de nombreux algorithmes de plongement ont été proposés pour différentes classes de graphes que nous mentionnerons en conclusion de cette section.

Algorithme 1.3 Petit-mondisation par plongement [197]

entrée(s) : Un graphe $H = (V, E)$ et un plongement σ de H dans $(\mathbb{R}^d, \ell_p)$
paramètre(s) : $\varepsilon > 0$
sortie(s) : Le graphe augmenté aléatoire $G = (V, E, F)$ avec $F = \{u \to L_u : u \in V\}$)
pour tout $u \in V$ **faire**
Tirer indépendamment un vecteur $\vec{\tau}_u \in \mathbb{R}^d$ suivant la distribution de probabilité :

$$\frac{d\Pr\{\vec{\tau}_u = \vec{\tau}\}}{d\vec{\tau}} = \frac{1}{Z} \cdot \frac{1}{(||\vec{\tau}||_p)^d \log^{1+\varepsilon}(e + ||\vec{\tau}||_p)}$$

où $Z = \int_{\vec{\tau} \in \mathbb{R}^d} \frac{d\vec{\tau}}{(||\vec{\tau}||_p)^d \log^{1+\varepsilon}(e+||\vec{\tau}||_p)}$ est la fonction de partition (et $e = \exp(1)$).

Poser $L_u := v$, où v est le nœud dont l'image $\sigma(v)$ par le plongement est la plus proche de $\sigma(u) + \vec{\tau}_u$ dans $\sigma(V)$ au sens de la norme ℓ_p.

Notre algorithme de petit-mondisation (algorithme 1.3) utilise un plongement donné avec le graphe pour tirer le contact distant de chaque nœud dans $\mathbb{R}^d$ suivant la loi harmonique de Kleinberg (légèrement modifiée pour converger sur tout $\mathbb{R}^d$) puis le reporte sur le nœud du graphe dont l'image par le plongement est la plus proche du point tiré au sort (fig. 1.5).

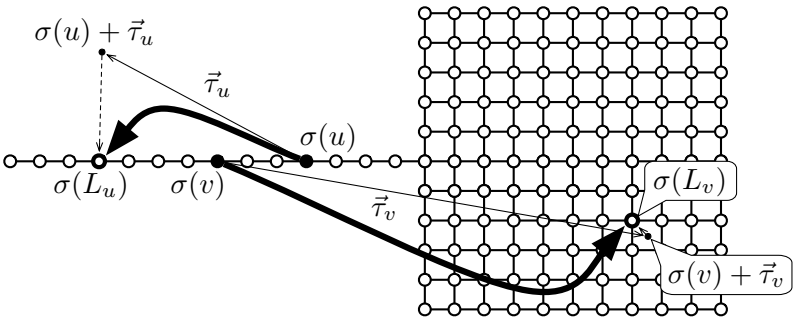


Figure 1.5 – Illustration de l'algorithme 1.3.

Théorème 1.12 ([194, 197]). *L'algorithme 1.3 est un schéma de petit-mondisation et augmente tout graphe $H = (V, E)$ à n sommets ayant un plongement de distorsion γ dans $(\mathbb{R}^d, \ell_p)$ en un petit-monde navigable $G = (V, E, F)$ où, avec probabilité $1 - \frac{1}{n}$, l'algorithme glouton décentralisé calcule des chemins de longueur $O(\gamma^d \log^{2+\varepsilon} n / \varepsilon)$ entre toutes les paires de sommets.*

Démonstration. Un simple calcul démontre que $Z \leqslant c_p \frac{1+e}{\varepsilon} \frac{2^d}{(d-1)!} < \infty$, où c_p est une constante qui ne dépend que de la norme ℓ_p. Analysons maintenant le routage glouton dans G. Supposons à présent que le message soit arrivé au nœud u en visant la cible t.

Remarquons que si $||(\sigma(u) + \vec{\tau}_u) - \sigma(t)||_p \leqslant \frac{||\sigma(u)-\sigma(t)||_p}{4\gamma}$, alors $\delta_H(L_u, t) \leqslant \frac{\delta_H(v,t)}{2}$. En effet, sous cette hypothèse,

$$\begin{aligned}
\delta_H(L_u, t) &\leqslant ||\sigma(L_u) - \sigma(t)||_p && \text{(définition du plongement)}\\
&\leqslant ||\sigma(L_u) - (\sigma(u) + \vec{\tau}_u)||_p + ||(\sigma(u) + \vec{\tau}_u) - \sigma(t)||_p && \text{(inégalité triangulaire)}\\
&\leqslant 2||(\sigma(u) + \vec{\tau}_u) - \sigma(t)||_p && (\sigma(L_u) \text{ est plus proche de } \sigma(u) + \vec{\tau}_u \text{ que } \sigma(t))\\
&\leqslant ||\sigma(u) - \sigma(t)||_p/2\gamma && \text{(par hypothèse)}\\
&\leqslant \delta_H(u, t)/2 && \text{(la distorsion de } \sigma \text{ vaut } \gamma)
\end{aligned}$$

Évaluons à présent la probabilité P que $\sigma(u) + \vec{\tau}_u$ tombe à distance au plus $\frac{||\sigma(u)-\sigma(t)||_p}{4\gamma}$ de $\sigma(t)$. En remarquant que

$$B_{\sigma(t),\ell_p}\left(\frac{||\sigma(u) - \sigma(t)||_p}{4\gamma}\right) \subset B_{\sigma(u),\ell_p}\left(\left(1 + \frac{1}{4\gamma}\right)||\sigma(u) - \sigma(t)||_p)\right),$$

et que les rayons de ces boules sont dans un rapport constant $(4\gamma + 1)$, on en déduit comme précédemment que

$$\begin{aligned}
P &\geqslant \frac{1}{Z} \frac{b_{\sigma(t),\ell_p}\left(\frac{||\sigma(u)-\sigma(t)||_p}{4\gamma}\right)}{\left((1 + \frac{1}{4\gamma})||\sigma(u) - \sigma(t)||_p\right)^d \log^{1+\varepsilon}\left(e + (1 + \frac{1}{4\gamma})||\sigma(u) - \sigma(t)||_p\right)}\\
&\geqslant \frac{c'_p\,\varepsilon}{d(5\gamma)^d \log^{1+\varepsilon}(2\gamma\delta_H(u, t) + e)},
\end{aligned}$$

où c'_p est une constante qui ne dépend que de d et de la norme ℓ_p. Ainsi à chaque pas, la distance courante à la cible est divisée par deux avec probabilité au moins $\Omega(\varepsilon/d(5\gamma)^d \log^{1+\varepsilon} n)$, indépendante des positions relatives de u et t. Un raisonnement similaire à celui du théorème 1.10 conclut via l'inégalité de Chernoff qu'avec probabilité $1 - \frac{1}{n}$, les longueurs des chemins calculés entre toutes les paires de nœuds sont toutes inférieures à $O(d(5\gamma)^d \log^{2+\varepsilon} n\,/\varepsilon)$. ■

Plongement et petit-mondisation. L'avantage de cette seconde méthode est son aspect purement géométrique qui repose uniquement sur l'existence d'un bon plongement. Par exemple, Abraham, Bartal et Neiman [2] proposent un algorithme de plongement randomisé de distorsion $O(\log^{1+\varepsilon} n)$ pour les graphes de dimension doublante α dans $(\mathbb{R}^{O(\alpha/\varepsilon)}, \ell_p)$. Combiné avec notre algorithme, nous obtenons une construction explicite pour le résultat de Slivkins [293] affirmant que les graphes de dimension doublante constante peuvent être augmentés en un petit monde. Nous démontrons de plus que l'ajout d'un seul lien par nœud suffit.

Au moment où nous avions obtenu ce résultat (2005), nous nourrissions beaucoup d'espoir de démontrer que la totalité des graphes de degré constant pourraient être "petit-mondisés" (au sens de la définition 1.8) par l'ajout d'un unique lien aléatoire par nœud tiré suivant une distribution bien choisie. Abraham et coll. [1] venaient en effet de démontrer que tout graphe admettait pour tout $\varepsilon > 0$ un plongement dans $(\mathbb{R}^{\log^2(\frac{1}{\varepsilon})}, \ell_1)$ où toutes sauf une fraction ε

des arêtes ont une distorsion de $O(\log(\frac{1}{\varepsilon}))$, c.-à-d. un plongement dans un espace de dimension constante et de distorsion constante sauf pour une fraction constante des arêtes. Aussi, nous pensions qu'il était probable que tout graphe puisse être petit-mondisé par cette méthode. D'autant que Fraigniaud [117] et Abraham et Gavoille [3] démontrèrent que d'autres classes de graphes aux métriques très différentes peuvent être augmentées en petit-mondes navigables : respectivement, les arbres de largeur arborescente[14] $O(\text{polylog}\, n)$ ou de cordalité[15] $O(\text{polylog}\, n)$, et les familles de graphes excluant un mineur.[16]

Par la suite, Fraigniaud, Lebhar et Loker [120] ont démontré qu'il existe une famille de graphes de dimension doublante $\delta(n)$ pour lesquels dès que $\delta(n) = \omega(\log\log n)$, quelle que soit la distribution de longs liens, il existe toujours (par un argument de comptage) une paire de nœuds pour laquelle l'algorithme glouton calcule un chemin de longueur $\Omega(n^{1/\sqrt{n}})$ dans le graphe augmenté. Il s'en suit que les graphes problématiques sont ceux ayant des métriques intermédiaires, entre polynomiales (de type euclidien, p. ex. les grilles) et arborescentes (p. ex., les expandeurs). Leur résultat démontre en particulier l'optimalité de notre schéma qui permet de petit-mondiser tous les graphes à croissance polylog n-bornée, et donc de dimension doublante $O(\log\log n)$ (théorème 1.10).

4 L'aspect dynamique

Dans cette section, nous nous intéressons à l'algorithme de routage et plus particulièrement à deux questions : l'algorithme glouton est-il le meilleur modèle du comportement des individus dans l'expérience de Milgram ? peut-on calculer des chemins de longueur optimale de façon décentralisée dans le graphe de Kleinberg pour la valeur $\alpha = d$ du paramètre ?

L'algorithme glouton proposé par Kleinberg ne nous a pas semblé correspondre à nos expériences personnelles de recherche sur le web. On se rend rapidement compte que de suivre le premier hyperlien qui semble nous rapprocher un peu plus de la page que l'on recherche, n'est pas très performant en pratique. On risque en particulier de se retrouver pris dans des culs-de-sac. Il nous a donc semblé plus naturel de considérer un algorithme qui s'informe autour de lui avant de décider à qui remettre le message. Notre but était donc à la fois d'obtenir une modélisation plus réaliste du comportement des utilisateurs et d'espérer obtenir une borne supérieure plus précise sur le diamètre du graphe de Kleinberg (qui ne sera établi qu'un an après la publication de nos travaux [206]).

[14] Une décomposition classique d'un graphe consiste à "l'aplatir" sous la forme d'un arbre de telle façon que les arêtes du graphe passent obligatoirement par les arêtes de l'arbre. Les sommets du graphes sont éventuellement dupliqués dans plusieurs nœuds de l'arbre qui peuvent contenir à leur tour plusieurs sommets du graphe. La *largeur arborescente* de cette décomposition est alors le nombre maximum de sommets du graphes présents dans un même nœud de l'arbre. La largeur arborescente ϑ (*treewidth*, en anglais) d'un graphe est alors le minimum des largeurs arborescentes de toutes les décompositions arborescentes de ce graphe. Introduit sous ce nom dans [269] pour démontrer la conjecture des familles de graphes closes par mineurs [270], ce paramètre s'est révélé être un paramètre pertinent en algorithmique pour séparer les instances par complexité croissante [74, 49] : de nombreux problèmes NP-difficiles admettent des algorithmes déterministes exacts de complexité $f(\vartheta)\cdot(n+m)$, (au moins) exponentielle en la largeur arborescente mais linéaire en la taille des données. Ces résultats reposent principalement sur la ressemblance des graphes de largeurs arborescentes faibles avec des arbres sur lesquels le paradigme diviser-pour-règner peut être appliqué avec succès.

[15] La *cordalité* (*chordality*, en anglais) d'un graphe est la longueur de son plus grand cycle sans corde. Ce paramètre mesure également la ressemblance d'un graphe à un arbre.

[16] On dit qu'un graphe H est un *mineur* d'un graphe G si on peut obtenir H à partir de G en appliquant une série d'opérations parmi les trois suivantes : éliminer un sommet, éliminer une arête, et contracter une arête. On dit qu'une famille de graphes exclut un mineur H si aucun des graphes de la famille n'admet H pour mineur.

Autres alternatives à l'algorithme glouton. Différentes tentatives dans ces directions ont été effectuées simultanément à la nôtre. Fraigniaud, Gavoille et Paul [119] ont démontré qu'en disposant de la connaissance de $\Theta(\log n)$ contacts distants d'autres nœuds autour de soi, une variante de l'algorithme glouton permet de calculer des chemins de longueur $\Theta(\log^{1+1/d} n)$. Le principe de cet algorithme est que $O(\log^{1/d} n)$ pas suffisent pour explorer $\log n$ nœuds dans une grille de dimension d, et donc pour diviser la distance à la cible par 2 avec probabilité constante. Ils démontrent de plus qu'aucun algorithme décentralisé disposant de ces informations supplémentaires ne peut avoir une latence inférieure.

D'autres ont étudié une variante du modèle de Kleinberg, appellée *modèle de percolation de Kleinberg*. Dans ce modèle, on ajoute un lien *entre chaque paire de nœuds* (u, v) avec probabilité $\propto 1/||u-v||^d$ indépendamment. Chaque nœud a alors en moyenne $k = \Theta(\log n)$ contacts distants. Coppersmith, Gamarnik et Sviridenko [73] ont démontré en 2002 que le diamètre de ce graphe est $\Theta(\log n \,/\, \log\log n)$. En proposant une version algorithmique de ce résultat, Manku, Noar et Wieder [204] ont démontré que l'algorithme décentralisé qui explore les longs liens des voisins des voisins du nœud courant calcule des chemins de longueurs optimales $O(\log n \,/\, \log\log n)$ avec une latence de $O(\log^3 n \,/\, \log\log n)$. Plus intéressant, ils démontrent également que, quelque soit la valeur de k, du simple fait que la probabilité d'un long lien décroît avec sa longueur, *aucun algorithme décentralisé ne peut calculer de chemins de routage avec une latence inférieure à celle de l'algorithme glouton*.[17] Cette propriété reste vraie dans le modèle standard de Kleinberg. Ainsi, tout algorithme décentralisé doit visiter au moins $\Theta(\log^2 n \,/k)$ nœuds dans le graphe augmenté $\mathcal{K}^d_{n,k}$.

4.1 Routage par explorations successives

Notre algorithme procède par explorations successives en évitant d'emprunter les longs liens qui nous éloignent de la cible, ou nous emmène trop près d'un contact déjà visité. Le principe est de réunir de la manière la plus compacte possible suffisamment de longs liens inexplorés, pour garantir que l'un d'eux nous rapproche beaucoup de la cible avec probabilité constante. Initialement, nous avions développé notre algorithme exclusivement pour la grille de Kleinberg $\mathcal{K}^d_{n,k}$. Cependant avec le recul, en relisant nos articles [195, 196], je me suis rendu compte que des arguments géométriques permettaient de simplifier et surtout étendre cet algorithme à l'ensemble des graphes petit-mondisés par les méthodes de la section précédente (algorithmes 1.2 et 1.3). Cette nouvelle présentation m'a de plus permis d'améliorer l'algorithme qui calcule à présent des chemins de longueur $O(\log n \log\log n)$ en espérance entre toutes les paires (pour les graphes à croissance bornée), au lieu de $O(\log n\,(\log\log n)^2)$ dans nos articles publiés à l'époque. Nous répondons ainsi positivement à l'un des dix problèmes ouverts posés par Kleinberg dans [190, problème ouvert n°3], qui s'interrogeait sur l'existence d'un algorithme décentralisé calculant en temps polylogarithmique des chemins de longueur $\ll \log n\,(\log\log n)^2$. La question de savoir si un tel algorithme décentralisé pourrait calculer des chemins de longueur optimale $O(\log n)$ reste cependant encore ouverte, mais nous pensons détenir maintenant des pistes pour l'attaquer. Nous en débattrons à la fin de cette section.

Remarque. Pour construire cet algorithme, nous nous sommes inspirés des techniques d'algorithmique d'approximation que j'avais mises en œuvre durant ma thèse. Notre objectif

[17] Bizarrement cette propriété qui est de loin la plus intéressante de l'article [204], n'est présentée que comme un lemme annexe.

étant de diminuer la longueur asymptotique du chemin calculé à une constante près, nous allons nous autoriser à perdre un peu sur la constante pour gagner beaucoup sur l'asymptotique.

Nous donnerons ici le détail des preuves car il s'agit d'un résultat nouveau qui n'a pas encore été publié. Par la suite, nous ne présenterons pas nos résultats avec le même niveau de détails et nous contenterons d'en expliquer les grands principes.

Familles de graphes considérées ici. Afin de simplifier la présentation de notre algorithme et de son analyse, nous nous limiterons dans ce mémoire au cas des graphes à croissance α-bornée, pour $\alpha > 1$ constant (définition 1.9). Notre analyse s'étend de manière directe à l'ensemble des graphes $O(\log^\rho r)$-bornée ou plongeables dans $(\mathbb{R}^d, \ell_p)$ avec distorsion γ. Les grilles d-dimensionnelles sont un exemple de graphe à croissance $O(1)$-bornée. Notre analyse s'applique donc aux graphes de Kleinberg $\mathcal{K}_{n,k}^d$ en particulier. Ce résultat est donc bien une généralisation et une amélioration de nos analyses précédentes [195, 196].

Nous considérerons donc à partir de maintenant une famille de graphes $\{H_n = (V_n, E_n)\}$ à croissance α-bornée où n désigne le nombre de sommets de H_n. Nous désignons par $\{G_n = (V_n, E_n, F_n)\}$ la famille de graphes aléatoires augmentés obtenue en appliquant à H_n le schéma de petit-mondisation 1.2 page 35.

Description générale. Notre algorithme (algorithme 1.4 page 44) procède par explorations successives jusqu'à être assez proche de la cible. Ensuite, il termine en appliquant le routage glouton de Kleinberg. Afin d'alléger les notations, nous noterons $\delta(\cdot,\cdot)$ la distance dans le graphe sous-jacent H_n. Par la suite, nous dirons qu'un contact v est un *bon contact* de u si v est strictement plus proche que u de la cible dans le graphe sous-jacent, *i.e.* si $\delta(v,t) < \delta(u,t)$. Nous dirons qu'un lien entre un nœud et un de ses bons contacts est un *bon lien*. Clairement, tout nœud (à l'exception de la cible) a toujours au moins un bon contact local.

La première partie de notre algorithme consiste en une suite de phases d'explorations. Nous allons démontrer qu'au terme de chaque phase d'exploration, le dernier niveau de l'arbre d'exploration contient avec probabilité constante $\Theta(\log^{1+\Theta(\varepsilon)} n)$ nœuds dont les contacts distants n'ont pas encore été explorés, et qu'avec probabilité constante, au moins un de ces contacts distants est $\Theta(\log^{\varepsilon} n)$ fois plus proche de la cible. Le gain par rapport à l'algorithme glouton sur la longueur du chemin obtenu provient du fait que tant que l'on est loin de la cible, la proportion de bons contacts est très importante et la phase d'exploration n'a pas besoin d'être profonde pour atteindre la largeur souhaitée. On gagne alors suffisamment sur les premières étapes d'exploration pour réduire la longueur du chemin obtenu, sans pour autant augmenter significativement la latence de l'algorithme (l'espérance du nombre total de nœuds visités).

Chaque phase d'exploration de notre algorithme consiste à ne suivre que des bons liens qui n'entrent pas en collision avec les zones déjà visitées. Pour cela, on se fixe deux paramètres : les profondeur et largeur maximales d'exploration $h_{\max}(r)$ et $b_{\max}$. Le principe est que l'on ne retient que les contacts distants à distance au moins $h_{\max}(r)$ des contacts distants déjà visités, afin d'assurer que la structure ne s'auto-intersecte pas. Remarquons que cette contrainte ne nous fait perdre qu'au plus un facteur constant sur la longueur du chemin et peut donc être faite sans dégrader l'asymptotique. Nous imposons également une largeur maximale d'exploration $b_{\max}$ afin de garantir que l'algorithme ne visite pas trop de nœuds en espérance. En effet, pour déterminer la valeur de la profondeur d'exploration $h_{\max}(r)$ garantissant que la largeur explorée est suffisante, nous n'utiliserons qu'une estimation à un facteur constant de la probabilité de non-recouvrement. L'erreur de cette estimation peut donc nous conduire à éventuellement surestimer la profondeur d'un facteur constant et donc éventuellement à explorer une puissance de la largeur souhaitée. Ainsi, une erreur bénigne d'un facteur constant sur la

Algorithme 1.4 Routage décentralisé par explorations successives – amélioration de [195, 196]

entrée(s) : un graphe $G = (V, E, F)$ augmenté à partir du graphe $H = (V, E)$ par le schéma 1.2 (page 35), une source s et une cible t.

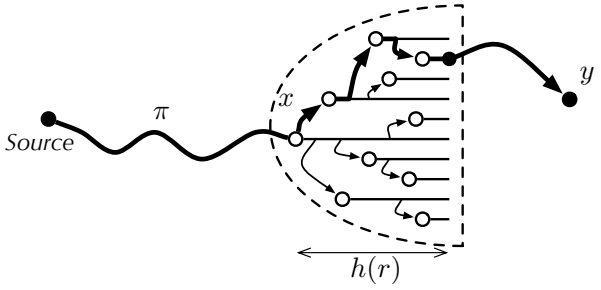
paramètre(s) : $\varepsilon > 0$, $h_{\max}(r)$, $b_{\max}$.
// les valeurs de $h_{\max}$ et $b_{\max}$ seront précisées aux équations (1.1) et (1.3) page 47.

initialisation : $u := s$.

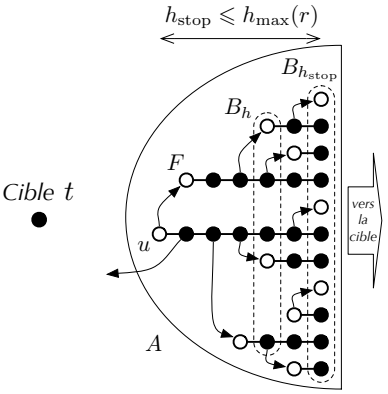
1. *1) Phases d'exploration*
2. Poser *now-go-greedy* $:= \left(\alpha^2 \cdot (h_{\max}(\log n))^{\log_2 \alpha} \cdot b_{\max}\right)^2 \cdot \log^{\varepsilon} n$.
3. **tant que** $\delta_H(u,t) >$ *now-go-greedy* **faire**
4. Poser $r := \delta_H(u,t)$ la distance de u à t dans le graphe de base H.
5. *1.a) Exploration (en largeur)*
6. Initialiser les ensembles B_0 et F à $\{u\}$, et $h := 1$.
7. **s'attendre à des exceptions**
8. **tant que** $h \leqslant h_{\max}(r)$ **faire**
9. Initialiser B_h à un ensemble vide ayant pour taille maximale $b_{\max}$ et lançant instantanément l'exception **est_plein!** si la taille de B_h atteint ou dépasse $b_{\max}$.
10. **pour chaque** $v \in B_{h-1}$ **faire**
11. Ajouter les bons contacts locaux de v à B_h.
12. **si** le contact distant L_v de v est bon et $\delta_H(L_v, F) \geqslant 2 \cdot h_{\max}(r)$ **alors**
13. Ajouter L_v à F et à B_h. *// où $\delta_H(w,F) =_{\text{def}} \min_{x \in F} \delta_H(w,x)$*
14. $h := h + 1$.
15. **en cas d'exception est_plein! faire** $h := h + 1$ et passer à la ligne 16.
16. Poser $h_{\text{stop}} := h - 1$ et $A := \bigcup_{h=0}^{h_{\text{stop}}} B_h$. *// par construction, $\#B_{h_{\text{stop}}} \leqslant b_{\max}$*
17. *1.b) Transmission du message*
18. Soit y, le nœud le plus proche de la cible (au sens de H) parmi les contacts locaux ou distants des nœuds de $B_{h_{\text{stop}}}$. Router le message le long du chemin de x à y dans l'arbre d'exploration A. Poser $x := y$.
19. *2) Phase finale (algorithme glouton de Kleinberg)*
20. Transmettre le message au contact local ou distant du sommet courant le plus proche de la cible t (au sens de H) jusqu'à atteindre la cible.

longueur asymptotique du chemin se révélerait avoir un coût prohibitif en terme de nombre de nœuds explorés. Nous avons donc rajouté dans l'algorithme un test d'arrêt de l'exploration dès que la largeur d'exploration est suffisante (émission de l'exception est_plein!). Nous gagnons ainsi sur les deux tableaux : nous pouvons nous contenter d'une surestimation à un facteur constant (donc "gratuite" asymptotiquement) de la profondeur d'exploration nécessaire, sans que cela ne nous coûte plus cher en terme de nombre de nœuds visités.

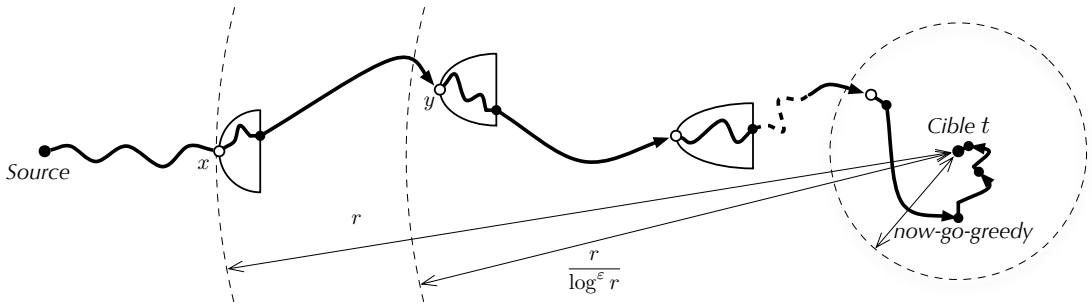
Principe général de l'analyse. La clé pour comprendre notre algorithme est que la distribution harmonique de Kleinberg dans la grille garantit que le contact distant d'un nœud u a la même probabilité, $1/\log_2 n$, d'appartenir à chacune des couronnes centrées sur u de rayon $[2^i, 2^{i+1})$, $C_i = \{w : 2^i \leqslant \delta(u,w) < 2^{i+1}\}$ pour $i = 1, \ldots, \log_2 n$. Ainsi, lorsque le message est à distance 2^{j+1} de la cible, les contacts distants visités ont donc chacun, j chances



(a) Prolongement du chemin π à la fin d'une étape d'exploration (étape 1.b de l'algorithme)



(b) L'arbre d'exploration (les nœuds en blanc sont ceux de F, c.-à-d. les têtes des chaînes de bons contacts locaux)



(c) Progression de l'algorithme : diminution de la distance à la cible d'un facteur $\log^{\varepsilon} r$ avec probabilité constante après chaque phase d'exploration pour $r >$ *now-go-greedy*, puis algorithme glouton pour $r \leqslant$ *now-go-greedy*.

Figure 1.6 – Principe de notre algorithme par explorations successives (algorithme 1.4).

sur $\log_2 n$ d'appartenir à l'une des j premières couronnes et donc d'être plus proches de la cible (v. le lemme 1.14 et la figure 1.7) et une chance sur $\log_2 n$ d'appartenir à la $j^{\text{ème}}$ et donc de réduire d'un facteur deux la distance à la cible. Ainsi, à distance 2^{j+1} de la cible, on peut espérer voir la structure explorée s'élargir d'un facteur $(1 + \frac{j}{\log_2 n})$ en moyenne chaque fois que la profondeur augmente de 1 jusqu'à ce que l'on atteigne une largeur de $\log n$ nœuds. On peut alors espérer avec probabilité constante qu'un de ces $\log n$ contacts distants soit à distance moitié de la cible. Pour atteindre une largeur $\log n$, il faut atteindre intuitivement une profondeur h telle que $(1 + \frac{j}{\log_2 n})^h = \log n$, c.-à-d. $h = \frac{\log \log n}{\log(1+\frac{j}{\log_2 n})} \approx \frac{\log_2 n \, \log \log n}{j}$. L'espérance de a longueur du chemin obtenu sera donc de l'ordre de :

$$\sum_{j=1}^{\log_2 n} \frac{\log_2 n \, \log \log n}{j} = O(\log n (\log \log n)^2).$$

Analyse de l'algorithme. La suite de cette section présente l'analyse rigoureuse de notre algorithme qui suit le principe général ci-dessus avec quelques différences notables qui nous permettront de nous affranchir d'un des facteurs $\log \log n$ sur la longueur du chemin final, améliorant ainsi nos résultats publiés dans [195, 196].

En reprenant la preuve du théorème 1.10 page 35, nous obtenons la majoration suivante de Z_u pour les graphes à croissance α-bornée.

Lemme 1.13. *Pour chaque graphe H_n et pour tout $u \in V_n$, $Z_u \leqslant \alpha \log_2 n$.*

Nous allons maintenant évaluer la probabilité de créer de nouvelles branches dans l'arbre en cours d'exploration.

Lemme 1.14 (Probabilité d'élargir la structure). *Étant donné un entier b, un sommet u à distance $r \geqslant b^2$ de la cible dans H_n, et un ensemble B d'au plus b nœuds à éviter situés à distance $< r$ de la cible, alors :*

$$\Pr\{\delta(L_u, t) < r \text{ et } L_u \notin B\} \geqslant \frac{\log r}{2\alpha^5 \log n}.$$

Démonstration. Considérons une géodésique[18] $\Gamma = (v_0 = u, v_1, \ldots, v_r = t)$ de u à t. Considérons les nœuds $u_i = v_{2^i}$ situés à distance 2^i de u sur Γ, pour $1 \leqslant i \leqslant \log_2 r$ (v. illustration figure 1.7). Remarquons que les boules $B_{u_i}(2^{i-2})$ sont toutes disjointes et incluses dans $B_t(r-1)$. On peut donc minorer la probabilité que le contact distant L_u de u soit

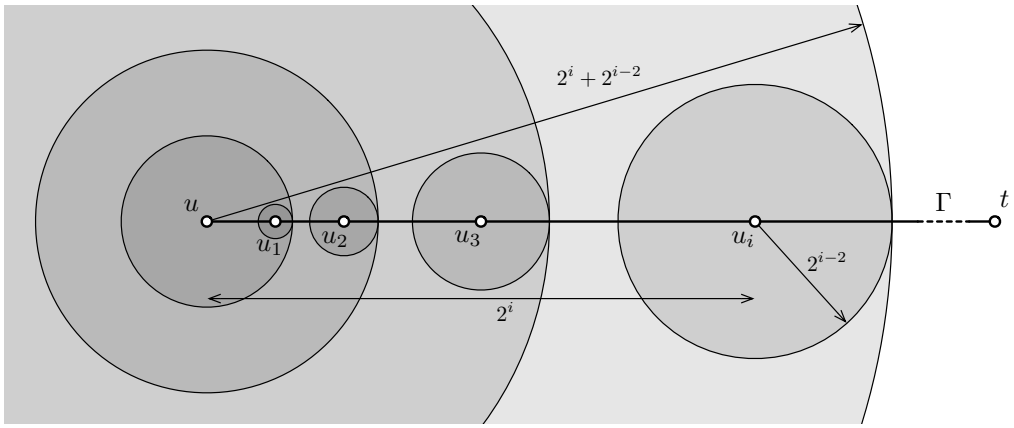


Figure 1.7 – Estimer la probabilité que le contact distant de u soit bon.

bon par la somme des probabilités qu'il tombe dans une de ces $\lfloor \log_2 r \rfloor$ boules. Or, comme $B_{u_i}(2^{i-2}) \subset B_u(2^i + 2^{i-2}) \subset B_{u_i}(2 \cdot 2^i + 2^{i-2})$ (v. figure 1.7),

$$\Pr\{L_u \in B_{u_i}(2^{i-2})\} \geqslant \frac{1}{Z_u} \frac{b_{u_i}(2^{i-2})}{b_u(2^i + 2^{i-2})} \geqslant \frac{1}{\alpha \log_2 n} \frac{b_{u_i}(2^{i-2})}{b_{u_i}((8+1)\,2^{i-2})} \geqslant \frac{1}{\alpha^5 \log_2 n}.$$

Reste à estimer la probabilité que L_u appartienne à l'ensemble B des nœuds à éviter. Remarquons que la probabilité que L_u soit à distance ℓ de u décroît avec ℓ, ainsi la probabilité que L_u appartienne à un ensemble B est maximisé lorsque les nœuds de B sont le plus proche possible de u. Nous obtenons donc un minorant de la probabilité que L_u soit bon et évite les sommets de B en supposant que ceux-ci sont situés dans la boule de rayon $b \leqslant \sqrt{r}$ autour de u, ainsi :

$$\Pr\{\delta(L_u, t) < r \text{ et } L_u \notin B\} \geqslant \sum_{i=\lceil \log_2 \sqrt{r} \rceil}^{\lfloor \log_2 r \rfloor} \frac{1}{\alpha^5 \log_2 n} \geqslant \frac{\log_2 r}{2\alpha^5 \log_2 n}. \qquad \blacksquare$$

[18] Une *géodésique* d'un graphe est un plus court chemin d'un point à un autre.

Nous pouvons donc à présent analyser l'algorithme 1.4. Commençons par remarquer que chaque phase d'exploration est disjointe des phases d'exploration précédentes. En effet, chaque phase explore des nœuds toujours strictement plus proches de la cible et commence à partir du nœud le plus proche de la cible atteint lors de la phase précédente. Nous pouvons donc analyser chaque phase d'exploration indépendamment des autres.

Fixons $\varepsilon > 0$ et posons $\dot{\alpha} = \log_2 \alpha$. Définissons les valeurs suivantes pour les paramètres $h_{\max}(r)$ et $b_{\max}$:

$$b_{\max} = \log^{1+\varepsilon\dot{\alpha}} n \tag{1.1}$$

$$\varrho_r = \frac{\log\left(\frac{r}{\log^{\varepsilon} n}\right)}{2\alpha^5 \log n} = \frac{\log r - \varepsilon \log\log n}{2\alpha^5 \log n} \tag{1.2}$$

$$h_{\max}(r) = \log_{(1+\varrho_r)} b_{\max}(r) = \Theta\left(\frac{(1+\varepsilon\dot{\alpha})\log\log n \log n}{\log r}\right) \tag{1.3}$$

Ainsi,

$$\begin{aligned} \textit{now-go-greedy} &= \left(\alpha^2 \left(h_{\max}(\log n)\right)^{\dot{\alpha}} b_{\max}\right)^2 \log^{\varepsilon} n \\ &= \frac{\alpha^4 (1+\varepsilon\dot{\alpha})^{2\dot{\alpha}} \log^{2+2\varepsilon\dot{\alpha}+\varepsilon} n \, (\log\log n)^{2\dot{\alpha}}}{\log^{2\dot{\alpha}}\left(1 + \frac{(1-\varepsilon)\log\log n}{2\alpha^5 \log n}\right)}. \end{aligned}$$

Sachant que $x \log 2 \leqslant \log(1+x) \leqslant x$ pour $x \in [0,1]$, nous obtenons l'encadrement suivant :

$$\textit{now-go-greedy} \;\leqslant\; \alpha^4 \left(2\alpha^5 \frac{1+\varepsilon\dot{\alpha}}{(1-\varepsilon)\log 2}\right)^{2\dot{\alpha}} \log^{2+2(1+\varepsilon)\dot{\alpha}+\varepsilon} n\,, \tag{1.4}$$

$$\text{et } \textit{now-go-greedy} \;\geqslant\; \alpha^4 \left(2\alpha^5 \frac{1+\varepsilon\dot{\alpha}}{1-\varepsilon}\right)^{2\dot{\alpha}} \log^{2+2(1+\varepsilon)\dot{\alpha}+\varepsilon} n \geqslant \log n \tag{1.5}$$

Considérons une phase d'exploration démarrant en u à distance $r >$ *now-go-greedy* de la cible t.

Lemme 1.15. *Pour tout h et pour tout $v \in B_h$, si $\delta(v,t) \geqslant \frac{r}{\log^{\varepsilon} n}$, alors au moment où l'on teste L_v (ligne 12 de l'algorithme 1.4) :*

$$\Pr\Big\{\delta(L_v,t) < \delta(v,t) \text{ et } \delta(L_v,F) > h_{\max}(r)\Big\} \geqslant \varrho_r =_{\text{def}} \frac{\log r - \varepsilon\log\log n}{2\alpha^5 \log n}.$$

Démonstration. Remarquons que F contient les points de départ des chaînes de contacts locaux de longueur au plus $h_{\max}(r)$ dans l'arbre d'exploration. La taille de F est donc la largeur de l'arbre exploré et $\#F \leqslant b_{\max}$. Si on note $B = \cup_{w\in F} B_w(h_{\max}(r))$, alors la probabilité que l'on souhaite estimer est :

$$\Pr\{\delta(L_v,t) < \delta(v,t) \text{ et } L_v \notin B\}$$

Or, $\frac{r}{\log^{\varepsilon} n} \geqslant (\#B)^2$, en effet : $\#B \leqslant \alpha^{1+\log_2(2\cdot h_{\max}(r))}\, \#F \leqslant \alpha^2\, h_{\max}(r)^{\dot{\alpha}}\, b_{\max}$ et comme $r \geqslant$ *now-go-greedy* $\geqslant \log n$, on a $h_{\max}(r) \leqslant h_{\max}(\log n)$ par décroissance de $h_{\max}(r)$, et

donc $\#B \leqslant \alpha^2\, h_{\max}(\log n)^{\dot\alpha}\, b_{\max} = \sqrt{\frac{\textit{now-go-greedy}}{\log^\varepsilon n}} \leqslant \sqrt{\frac{r}{\log^\varepsilon n}}$. Ainsi $\delta(v,t) \geqslant (\#B)^2$ et le lemme 1.14 conclut que :

$$\Pr\{\delta(L_v,t) < \delta(v,t) \text{ et } L_v \notin B\} \geqslant \frac{\log \delta(v,t)}{2\alpha^5 \log n} \geqslant \frac{\log\left(\frac{r}{\log^\varepsilon n}\right)}{2\alpha^5 \log n} = \varrho_r. \qquad \blacksquare$$

Nous en déduisons que tant que les nœuds explorés restent à une distance supérieure à $\frac{r}{\log^\varepsilon n}$ de la cible, la probabilité d'élargir l'arbre exploré avec un nouveau contact distant est d'au moins $\frac{\log r - \varepsilon \log\log n}{2\alpha^5 \log n}$. Remarquons que ce lemme signifie que tant que l'on est à distance supérieure à *now-go-greedy* de la cible, on peut négliger (ou plutôt absorber) la probabilité de collision interne de l'arbre exploré, au sens où cette probabilité ne diminue que d'un facteur constant sa probabilité de croissance, ce qui n'aura pas d'impact sur la longueur asymptotique du chemin calculé.

Nous dirons que la phase d'exploration est un *succès* si au moins un des deux événements suivants se produit : *A)* un sommet à distance au plus $\frac{r}{\log^\varepsilon n}$ de la cible est exploré durant la phase 1.a) de l'algorithme, ou bien *B)* $\#B_{h_{\text{stop}}} \geqslant \frac{b_{\max}}{2}$, c.-à-d. le dernier niveau de l'arbre exploré contient au moins $b_{\max}/2$ nœuds dont les contacts distants n'ont pas encore été regardés.

Tant que l'événement *A)* ne se produit pas, le contact distant de chaque sommet visité est ajouté à l'arbre des nœuds explorés par l'algorithme avec une probabilité $\geqslant \varrho_r$. Supposons à présent l'événement *A)* se produit. Considérons les parties de la structure arborescente explorée par l'algorithme qui se situent à une distance $\leqslant \frac{r}{\log^\varepsilon n}$ de la cible, et remplaçons-les artificiellement par un arbre aléatoire où chaque nœud a un ou deux fils avec probabilité $1-\varrho_r$ et ϱ respectivement. Un simple argument de couplage stochastique démontre que la probabilité de l'événement *"A) ou B)"* est supérieure à celle que le niveau $h_{\max}(r)$ de la structure modifiée atteigne une largeur au moins $b_{\max}/2$.

Considérons à présent le processus de Galton-Watson[19] $\mathcal{GW}_{\varrho_r}$ qui, partant d'un sommet unique, construit un arbre aléatoire en donnant à chaque sommet du niveau courant un ou deux fils avec probabilités $1-\varrho_r$ et ϱ_r respectivement. Le cardinal de chaque niveau de l'arbre $\mathcal{GW}_{\varrho_r}$ est dominé stochastiquement par le cardinal des niveaux de la structure artificielle décrite ci-dessus. Ainsi la probabilité de l'événement *"A) ou B)"* est supérieure à celle que l'arbre $\mathcal{GW}_{\varrho_r}$ atteigne une largeur au moins $b_{\max}/2$ au niveau $h_{\max}(r)$.

Notons b_h la variable aléatoire du nombre de nœuds au niveau h dans le processus $\mathcal{GW}_{\varrho_r}$. En estimant l'espérance et la variance de b_h, nous obtenons par la méthode du second moment le minorant suivant sur la probabilité de succès d'une phase d'exploration.

Lemme 1.16 (Processus de Galton-Watson [195, 196]). *Nous avons :*

$$\mathbb{E}[b_h] = (1+\varrho_r)^h \quad \textit{et} \quad \mathbb{E}[(b_h)^2] \leqslant \left(1 + \frac{1-\varrho_r}{1+\varrho_r}\right) \cdot (1+\varrho_r)^{2h}.$$

Considérons une variable aléatoire X à valeurs réelles et notons $X^+ = \max(0, X)$ et $X^- = \max(0, -X)$ telles que $X = X^+ - X^-$. Nous pouvons généraliser aux variables à valeurs négatives la méthode du second moment proposée à la remarque 3.1 de [173] .

[19] Pour une introduction aux processus de Galton-Watson, se référer à l'excellent cours de Brigitte Chauvin sur les martingales [64] à la non moins remarquable école ALÉA 2002.

Lemme 1.17 (Méthode du second moment [195, 196]).
Si $\mathbb{E}[X^-] \leqslant 2\,\mathbb{E}[X^+]$, *alors* $\Pr\{X > 0\} \geqslant \mathbb{E}[X]^2/\mathbb{E}[X^2]$.

En appliquant cette inégalité à la variable aléatoire $X = b_h - \mathbb{E}[b_h]/2$, nous obtenons :

Corollaire 1.18 (Largeur de $\mathcal{GW}_{\varrho_r}$). *Pour tout* h, $\Pr\{b_h \geqslant \mathbb{E}[b_h]/2\} \geqslant \frac{1}{5}$.

Or, par définition de $b_{\max}$ et $h_{\max}(r)$ (page 47),

$$h_{\max}(r) = \log_{1+\varrho_r} b_{\max} \quad \text{d'où} \quad \mathbb{E}[b_{h_{\max}(r)}] = (1+\varrho_r)^{h_{\max}(r)} = b_{\max}.$$

Nous en déduisons donc via notre couplage que chaque phase d'exploration est un succès avec probabilité au moins $\frac{1}{5}$.

Proposition 1.19 (Probabilité de succès d'une phase d'exploration). *À la fin de toute phase d'exploration démarrée depuis un nœud* u *à distance* $r >$ *now-go-greedy de la cible, avec probabilité au moins* $\frac{1}{5}$, *soit un sommet à distance* $\leqslant \frac{r}{\log^\varepsilon n}$ *de la cible a été visité, soit* $\#B_{h_{\text{stop}}} \geqslant \frac{b_{\max}}{2}$.

La proposition suivante va nous permettre de conclure l'analyse de notre algorithme.

Proposition 1.20. *Soit* B *un ensemble de* $\frac{\log^{1+\varepsilon\dot\alpha} n}{2}$ *nœuds à distance au plus* r *de la cible. La probabilité que le contact distant d'un des nœuds de* B *se situe à une distance au plus* $r/\log^\varepsilon n$ *de la cible, vaut au moins* $1 - 2^{-1/2\alpha^4} > 0$, *constante indépendante de* r *et* n.

Démonstration. Soit $v \in B$. Comme $B_t(\frac{r}{\log^\varepsilon n}) \subseteq B_v(r + \frac{r}{\log^\varepsilon n}) \subseteq B_t(2r + \frac{r}{\log^\varepsilon n})$, la probabilité que son contact distant L_v se situe à distance $\leqslant r/\log^\varepsilon n$ de la cible est minorée par :

$$\begin{aligned}\frac{1}{Z_v}\frac{b_t(\frac{r}{\log^\varepsilon n})}{b_v(r+\frac{r}{\log^\varepsilon n})} &\geqslant \frac{1}{Z_v}\frac{b_t(\frac{r}{\log^\varepsilon n})}{b_t(2r+\frac{r}{\log^\varepsilon n})} \geqslant \frac{1}{\alpha\log_2 n}\,\alpha^{-\lceil\log_2(1+2\log^\varepsilon n)\rceil} \\ &\geqslant \frac{1}{\alpha^{2+\log_2(3\log^\varepsilon n)}\log_2 n} \geqslant \frac{\log 2}{\alpha^4\log^{1+\varepsilon\dot\alpha} n}.\end{aligned}$$

Il s'en suit que la probabilité que le contacts distant d'un des nœuds de B soit à distance au plus $r/\log^\varepsilon n$ de la cible est minorée par :

$$\begin{aligned}1 - \left(1 - \frac{\log 2}{\alpha^4\log^{1+\varepsilon\dot\alpha} n}\right)^{\frac{\log^{1+\varepsilon\dot\alpha} n}{2}} &= 1 - \exp\left(\frac{\log^{1+\varepsilon\dot\alpha} n}{2}\log\left(1 - \frac{\log 2}{\alpha^4\log^{1+\varepsilon\dot\alpha} n}\right)\right) \\ &\geqslant 1 - 2^{-1/2\alpha^4}.\end{aligned}$$

■

Corollaire 1.21. *À la fin de toute phase d'exploration démarrée en un nœud* u *à distance* $r >$ *now-go-greedy de la cible, la probabilité que le nœud* y *vers lequel le message est routé (ligne 18) se situe à distance au plus* $r/\log^\varepsilon n$ *de la cible vaut au moins* $\varsigma =_{\text{def}} (1 - 2^{-1/2\alpha^4})/5 > 0$, *constante indépendante de* r *et* n.

Nous pouvons maintenant conclure l'analyse de notre algorithme de routage.

Théorème 1.22 (Amélioration de [195, 196]). *L'algorithme décentralisé 1.4 calcule des chemins de longueur quasi-optimale $O(\log n \log\log n\,/\varepsilon)$ en espérance entre toutes les paires de nœuds, avec une latence quasi-optimale $O(\log^{2+O(\varepsilon)} n\,/\varepsilon)$.*

De plus, avec probabilité $1-O(\frac{1}{n})$, à l'instar de l'algorithme glouton, la longueur du plus long chemin calculé entre toutes les paires est inférieure à $O(\log^2 n\,/\varepsilon)$.

(les $O(\cdot)$ s'entendent pour $n\to\infty$ et $\varepsilon\to 0$.)

Démonstration. Afin d'obtenir la borne désirée sur la longueur des chemins entre *toutes* les paires de nœuds simultanément avec forte probabilité, nous allons analyser les phases 1) et 2) de l'algorithme au travers d'une unique variable aléatoire.

Considérons tout d'abord la seconde étape de l'algorithme. Nous savons par l'analyse faite au théorème 1.10, qu'à chaque pas de l'algorithme glouton 1.1, la probabilité que le contact distant soit à distance moitié de la cible est supérieure à $\frac{1}{c\log n}$ pour une certaine constante $c>0$ et ce indépendamment de la distance courante à la cible. Ainsi, comme l'indique Kleinberg [188], la probabilité de trouver un contact distant à distance moitié de la cible parmi $\log n$ nœuds est supérieure à une certaine constante $\varsigma'>0$ définie comme suit :

$$1-\left(1-\frac{1}{c\log n}\right)^{\log n}\geqslant 1-e^{-1/c}=_{\text{def}}\varsigma'>0.$$

La longueur du chemin calculé à partir d'un nœud à distance $r\leqslant$ *now-go-greedy* par notre algorithme est donc majorée stochastiquement par la variable :

$$X=\sum_{i=1}^{\lceil\log_2(\textit{now-go-greedy})\rceil} X_i\,\log n,$$

où les X_i sont des variables aléatoires géométriques i.i.d. définies par

$$\Pr\{X_i=t\}=\varsigma'(1-\varsigma')^{t-1}\quad\text{pour tout } t\in\mathbb{N}^*.$$

Chaque X_i représente un majorant du nombre de groupes de $\log n$ nœuds visités dans la couronne de rayon $(2^{i-1},2^i]$ centrée sur la cible, avant de rencontrer un contact distant deux fois plus proche de la cible.

De la même façon, nous savons par le corollaire 1.21 que lors de la première étape de l'algorithme, la distance du message à la cible est indépendamment divisée par $\log^\varepsilon n$ après chaque phase d'exploration avec probabilité au moins ς. La longueur du chemin calculé jusqu'à ce que le message arrive à distance $\leqslant$ *now-go-greedy* de la cible, est donc dominée stochastiquement par la variable aléatoire suivante :

$$Y=\sum_{i=\lfloor\log_{\log^\varepsilon n}(\textit{now-go-greedy})\rfloor}^{\lceil\log_{\log^\varepsilon n} n\rceil} Y_i\cdot h_{\max}(\log^{i\varepsilon} n),$$

où les Y_i sont des variables aléatoires géométriques i.i.d. définies par

$$\Pr\{Y_i=t\}\;=\varsigma(1-\varsigma)^{t-1}\quad\text{pour } t\in\mathbb{N}^*.$$

Chaque Y_i représente un majorant du nombre aléatoire de phases d'exploration de profondeur au plus $h_{\max}(\log^{i\varepsilon} n)$ exécutées dans chaque couronne de rayon $(\log^{i\varepsilon} n,\log^{(i+1)\varepsilon} n]$ centrée sur la cible t.

En utilisant que $\log(1+x) \geqslant x \log 2$ pour $x \in [0,1]$, nous avons

$$h_{\max}(\log^{i\varepsilon} n) = \frac{(1+\varepsilon\dot{\alpha}) \log\log n}{\log\left(1 + \frac{(i-1)\varepsilon \log\log n}{2\alpha^5 \log n}\right)} \leqslant \frac{2\alpha^5 (1+\varepsilon\dot{\alpha}) \log n}{(i-1)\,\varepsilon \log 2}.$$

Comme $\log_{\log^\varepsilon n} n \leqslant \log n$ et d'après les inégalités (1.4) et (1.5) (page 47), il existe deux constantes C et C' qui ne dépendent que de α telles que :

$$\log\log n \leqslant \log_2(\textit{now-go-greedy}) \leqslant (C + C'\varepsilon)\ \log\log n.$$

Nous obtenons donc que la longueur totale du chemin calculé par notre algorithme est dominée stochastiquement par la variable aléatoire Z :

$$Z = \sum_{i=1}^{\lceil (C+C'\varepsilon)\ \log\log n \rceil} X_i \cdot \log n + \sum_{i=\lfloor \log\log n \rfloor}^{\lceil \log n \rceil} Y_i \cdot \frac{C''}{(i-1)\,\varepsilon}\ \log n,$$

pour une certaine constante C'' qui ne dépend que de α. On obtient ainsi un majorant de l'espérance de la longueur du chemin calculé :

$$\begin{aligned}\mathbb{E}[Z] &= \frac{\lceil (C+C'\varepsilon) \log\log n \rceil}{\varsigma'} \log n + \frac{C''}{\varsigma\varepsilon} \left(H_{\lceil \log n \rceil - 1} - H_{\lfloor \log\log n \rfloor - 2}\right) \log n \\ &= \Theta\left(\frac{1}{\varepsilon}\ \log n\ \log\log n\right) \text{ pour } n \to \infty \text{ ou } \varepsilon \to 0,\end{aligned}$$

où $H_x =_{\text{def}} \sum_{i=1}^{x} \frac{1}{i}$ désigne la somme harmonique qui vérifie : $\log(x+1) < H_x < 1 + \log x$. L'espérance de la longueur du chemin calculé est donc $O(\frac{1}{\varepsilon} \log n\ \log\log n)$ pour toute paire de nœuds, contre $\Theta(\log^2 n)$ pour l'algorithme glouton de Kleinberg.

Remarquons cependant que pour tout $\eta > 0$,

$$\begin{aligned}\Pr\{X_i \geqslant \eta \log n\} &= \sum_{t \geqslant \eta \log n} (1-\varsigma')^{t-1}\varsigma' \approx (1-\varsigma')^{\eta \log n} = \exp\left(\eta \log n \log(1-\varsigma')\right) \\ &= \frac{1}{n^{\Theta(\eta)}} \text{ pour } \eta \to 0 \text{ ou } n \to \infty.\end{aligned}$$

Ainsi, nous ne pouvons pas espérer démontrer que la longueur du plus long chemin calculé entre deux points soit de l'ordre de $o(\log^2 n)$ avec une probabilité $o(\frac{1}{n^\eta})$ quelque soit $\eta > 0$. Nous ne pensons pas que cela soit dû aux approximations faites durant l'analyse puisqu'elles n'induisent qu'une erreur d'un facteur constant sur les estimations des probabilités ς et ς', mais que ceci est bien dû à l'utilisation de l'algorithme glouton pour les dernières étapes de l'algorithme. Notre algorithme ne diminue donc pas significativement (sauf surprise) la longueur du *plus long* chemin calculé entre la pire paire de nœuds par rapport à l'algorithme glouton.

Nous pouvons en revanche garantir qu'il n'en calcule jamais de pire. Pour cela nous utilisons le théorème IX.7 du volume n°2 de la bible de l'analyse d'algorithmes [116, p. 643] que m'a aimablement indiqué Lucas Gerin (merci !). L'idée est de calculer la transformée de

Laplace de Z pour obtenir une borne exponentielle sur le poids de la queue de sa distribution. Considérons $\lambda(s) = \mathbb{E}\left[e^{sZ}\right]$. Le principe est de remarquer que si $\lambda(s) < \infty$ pour tout $s \in [0, a]$ avec $a > 0$, alors :

$$\Pr\{Z \geqslant z\} = \Pr\left\{e^{sZ} \geqslant e^{sz}\right\} = \Pr\left\{e^{sZ} \geqslant \frac{e^{sz}}{\lambda(s)}\mathbb{E}\left[e^{sZ}\right]\right\} \leqslant \lambda(s)\, e^{-sz},$$

par l'inégalité de Markov. Ainsi pour $s = a$, on obtient que $\Pr\{Z \geqslant z\} \leqslant \lambda(a)\, e^{-az}$. Étudions donc le rayon de convergence de $\mathbb{E}\left[e^{sZ}\right]$. Comme les X_i et Y_i sont tous indépendants,

$$\begin{aligned}\mathbb{E}\left[e^{\frac{sZ}{\log n}}\right] &= \prod_{i=1}^{\lceil (C+C'\varepsilon)\,\log\log n\rceil} \mathbb{E}\left[e^{sX_i}\right] \cdot \prod_{i=\lfloor \log\log n\rfloor - 1}^{\lceil \log n\rceil - 1} \mathbb{E}\left[e^{sC''Y_i/i\varepsilon}\right] \\ &= \left(\mathbb{E}\left[e^{sX_1}\right]\right)^{(C+C'\varepsilon)\log\log n+1} \cdot \prod_{i=\lfloor \log\log n\rfloor - 1}^{\lceil \log n\rceil - 1} \mathbb{E}\left[e^{sC''Y_1/i\varepsilon}\right]\end{aligned}$$

Or, pour $s < -\log(1-\varsigma')$, on a $(1-\varsigma')e^s < 1$ et

$$\mathbb{E}\left[e^{sX_1}\right] = \sum_{t\geqslant 1} \varsigma'(1-\varsigma')^{t-1}e^{st} = \varsigma' e^s \sum_{t\geqslant 0}((1-\varsigma')e^s)^t = \frac{\varsigma' e^s}{1-(1-\varsigma')e^s} < \infty.$$

Posons $\gamma_i = C''/i\varepsilon$. De même, pour $s < -\frac{\varepsilon}{C''}\log(1-\varsigma) \leqslant -\frac{1}{\gamma_i}\log(1-\varsigma)$, nous avons $(1-\varsigma)e^{s\gamma_i} < 1$ et

$$\begin{aligned}\mathbb{E}\left[e^{s\gamma_i Y_1}\right] &= \sum_{t\geqslant 1} \varsigma(1-\varsigma)^{t-1}e^{s\gamma_i t} = \varsigma e^{s\gamma_i} \sum_{t\geqslant 0}((1-\varsigma)e^{s\gamma_i})^t \\ &= \frac{\varsigma e^{sC''/i\varepsilon}}{1-(1-\varsigma)e^{sC''/i\varepsilon}} \leqslant \frac{\varsigma e^{sC''/i\varepsilon}}{1-(1-\varsigma)e^{sC''/\varepsilon}} < \infty.\end{aligned}$$

Posons $\gamma = \min\{-\log(1-\varsigma'), -\frac{\varepsilon}{C''}\log(1-\varsigma)\}$. On a $e^{\gamma/2} \leqslant \frac{1}{\sqrt{1-\varsigma'}}$ et $e^{\gamma C''/2\varepsilon} \leqslant \frac{1}{\sqrt{1-\varsigma}}$. Ainsi, pour $s = \gamma/2$,

$$\begin{aligned}\mathbb{E}\left[e^{\frac{\gamma}{2}Z/\log n}\right] &\leqslant \left(\frac{\varsigma'}{(1-\sqrt{1-\varsigma'})\sqrt{1-\varsigma'}}\right)^{(C+C'\varepsilon)\log\log n+1} \cdot (1-\varsigma)^{-H_{\log n}/2} \\ &\qquad \cdot \left(\frac{\varsigma}{1-\sqrt{1-\varsigma}}\right)^{\log n} \\ &\leqslant n^D,\end{aligned}$$

pour une certaine constante D indépendante de n et ε. Par application de l'inégalité de Markov à la transformée de Laplace de Z, nous obtenons donc que

$$\Pr\{Z > A\cdot\log^2 n\} \leqslant \mathbb{E}\left[e^{\frac{\gamma}{2}Z/\log n}\right] e^{-\frac{\gamma}{2}A\log n} \leqslant n^D e^{-\frac{\gamma}{2}A\log n} \leqslant \frac{1}{n^3}$$

pour tout $A \geqslant 2(3+D)/\gamma = O(1/\varepsilon)$ pour $\varepsilon \to 0$. En sommant ces probabilités sur l'ensemble des $n(n-1)$ paires de nœuds, nous en concluons que la probabilité que le chemin le plus long calculé par notre algorithme soit supérieure à $B\log^2 n/\varepsilon$, est inférieure à $\frac{1}{n}$ pour une certaine constante B.

Étudions maintenant la latence de l'algorithme, c.-à-d. l'espérance du nombre de nœuds visités pour calculer un chemin entre une paire de nœuds. Chaque phase d'exploration de l'algorithme depuis un nœud à distance $r \geqslant$ *now-go-greedy* de la cible, visite au plus $b_{\max} \cdot h_{\max}(r)$ nœuds. Ainsi, de même que précédemment, la latence est stochastiquement dominée par la variable aléatoire :

$$L = \sum_{i=1}^{\lceil (C+C'\varepsilon)\,\log\log n\rceil} X_i \cdot \log n + \sum_{i=\lfloor \log\log n\rfloor}^{\lceil \log n\rceil} Y_i \cdot \log^{1+\varepsilon\dot{\alpha}} n \cdot \frac{C''}{(i-1)\,\varepsilon}\,\log n.$$

dont l'espérance vaut :

$$\mathbb{E}[L] = O(\log^{2+\varepsilon\dot{\alpha}} n\,\log\log n\,/\varepsilon) = O(\log^{2+O(\varepsilon)} n\,/\varepsilon).$$ ■

Lorsque le nombre de contacts distants k n'est pas constant. Dans nos articles originaux [195, 196], nous avions également proposé un algorithme calculant des chemins dont la longueur décroît en espérance avec k, précisément $O(\log n\,(\log\log n\,/\,\log k)^2)$ obtenu en modulant la profondeur d'exploration en fonction de k, pour tout $k = O(\log n)$. Je n'ai pas jugé utile de compliquer l'analyse ici en autorisant k à varier avec n, mais il est clair que l'on peut espérer le même genre de gain en modifiant la fonction $h_{\max}(r)$ pour tirer parti d'un k variable.

Peut-on faire mieux de façon décentralisée avec une latence polylogarithmique ? C'est la question que Kleinberg posait au problème ouvert n°3 dans [190] suite à nos publications [195, 196] démontrant l'existence d'un algorithme calculant des chemins de longueur $O(\log n\,(\log\log n)^2)$ en espérance avec une latence optimale $O(\log^2 n)$. Le résultat présenté ici, basé sur une compréhension plus géométrique de notre algorithme précédent, démontre que Oui. Notre nouvel algorithme calcule des chemins de longueur $O(\log n\,\log\log n)$ en espérance avec une latence quasi-optimale $O(\log^{2+\varepsilon} n)$. Ce gain est obtenu en poussant simplement l'exploration un peu plus loin, sans surcoût asymptotique, de sorte à obtenir un arbre d'une largeur de $\log^{1+\Theta(\varepsilon)} n$ au lieu de $\log n$ comme dans nos articles originaux. Ceci permet de se rapprocher de la cible d'un facteur $\log^{\varepsilon} n$, au lieu de 2, avec probabilité constante après chaque exploration. Nous gagnons ainsi un facteur $\log\log n$ sur l'espérance de la longueur totale. Nous savons à présent par [206] que le diamètre du graphe de Kleinberg est $\Theta(\log n)$. Pourtant, je suis à peu près convaincu que l'on ne peut pas faire mieux qu'une longueur $\Theta(\log n\log\log n)$ en espérance de façon décentralisée en ne visitant qu'un nombre polylogarithmique de nœuds. La raison en est que la visite d'un nombre polylogarithmique de nœuds permet de se rapprocher d'un facteur $O(\log^{O(1)} n)$ de la cible avec probabilité constante mais pas plus. Il serait très intéressant de rendre rigoureuse cette analyse (si elle est exacte). Cela établirait ainsi l'existence d'un saut[20] entre le diamètre du graphe et la longueur minimale des chemins calculables efficacement par un algorithme décentralisé utilisant uniquement la géométrie sous-jacente. Ceci démontrerait en particulier que le diamètre n'est pas un paramètre statistique pertinent pour l'analyse des graphes sociaux.[21]

[20] *Gap*, en anglais.

[21] Avec George Giakkoupis nous avons démontré par la suite dans [134] que l'on ne peut effectivement pas faire mieux que $\Omega(\log n\log\log n)$ en dimension 1 en ne visitant qu'un nombre polylogarithmique de nœuds, mais que l'on peut trouver des chemins de longueur optimale $O(\log n)$ dès que la dimension de la grille sous-jacente est $\geqslant 2$.

4.2 Les gens procèdent-ils par algorithme ? et si oui le(s)quel(s) ?

Durant la thèse d'Emmanuelle Lebhar, nous nous sommes posés la question de déterminer comment procèdent les individus pour transmettre les messages dans l'expérience de Milgram. Nous avons recherché dans ce but un paramètre statistique dont la valeur varie significativement d'un algorithme de routage à l'autre, et qui soit facile à mesurer à partir des données de l'expérience. Nous avons proposé d'étudier trois paramètres :

- la probabilité qu'un nœud soit visité pendant l'exécution de l'algorithme.
- la distribution de la charge des liens en fonction de leur longueur : si on note $\mathcal{C}$ l'ensemble aléatoire des $n(n-1)$ chemins calculés par l'algorithme entre toutes les paires de nœuds, il s'agit de l'espérance du nombre de chemins de $\mathcal{C}$ qui passent par un lien d'une longueur donnée. Dans le cas de l'expérience de Milgram, la longueur d'un lien correspondrait à la distance (en kilomètres terrestres) entre les positions de l'expéditeur et du destinataire à chaque transmission.
- l'espérance du taux de liens locaux sur le chemin calculé en fonction de la distance de la source à la cible.

Dans le troisième chapitre de sa thèse [194], Emmanuelle Lebhar a pu obtenir les équations permettant de calculer *exactement* ces quantités dans le cas de l'anneau de Kleinberg $\mathcal{K}^1_{n,1}$ pour deux algorithmes principalement :

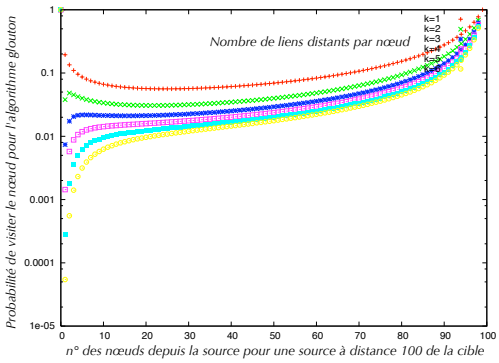
- l'algorithme glouton de Kleinberg (algorithme 1.1) ;
- l'algorithme par exploration *locale* (algorithme 1.5) qui regarde les contacts distants des ω bons contacts locaux du nœud courant, puis sélectionne le plus proche de la cible pour lui transmettre le message (algorithme du type de celui analysé dans [119]).

Algorithme 1.5 Algorithme par exploration locale d'une fenêtre de taille ω

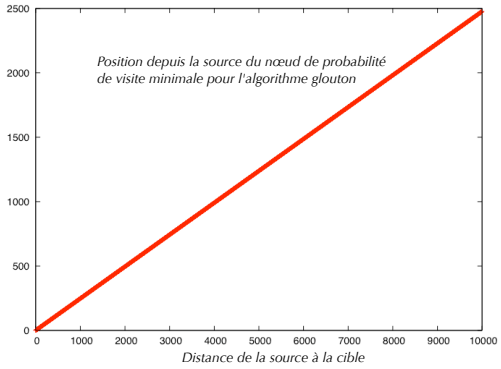
entrée(s) : L'anneau augmenté aléatoirement de Kleinberg $\mathcal{K}^{d=1}_{n,k=1}$, une source s et une cible t.
paramètre(s) : la taille de la fenêtre $\omega \in \mathbb{N}^*$.
initialisation : $x := s$.
tant que $x \neq t$ **faire**
Soit $F(x, \omega, t)$ l'ensemble des nœuds $\{x, x+1, \ldots, x+\omega-1\}$ si $x < t$, et $\{x, x-1, \ldots, x-\omega+1\}$ sinon.
Soit y le nœud le plus proche de la cible parmi les contacts distants ou locaux des nœuds de $F(x, \omega, t)$. Router le message vers y via $F(x, \omega, t)$. Poser $x := y$.

Les équations de récurrence obtenues pour ces paramètres se sont malheureusement révélées avoir des comportements étonnamment compliqués que nous n'avons pas su analyser précisément. En fait nous avons appris par la suite des mathématiciens que nous avons consultés, que ce type d'équations de récurrence est très difficile à analyser mathématiquement et que même des propriétés comme la monotonie sont le plus souvent inaccessibles.

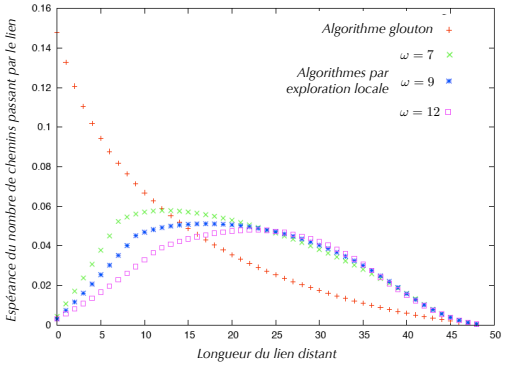
Emmanuelle Lebhar a en revanche pu en tracer les courbes, représentant les valeurs *exactes* des probabilités et espérances évaluées (figure 1.8 page 55) :



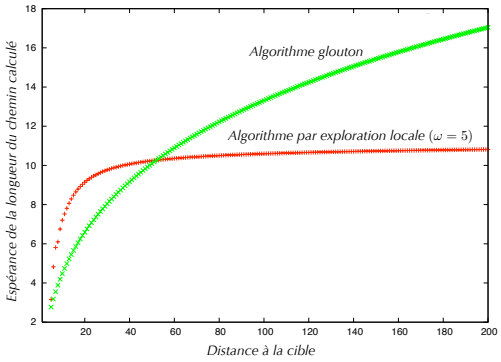
(a) Probabilité de visite d'un nœud entre la source et la cible pour l'algorithme glouton en fonction du nombre k de contacts distants par nœud. *(échelle semi-log)*



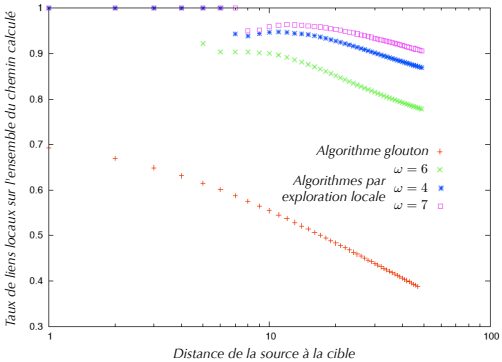
(b) Position du nœud de probabilité de visite minimale entre la source et la cible pour l'algorithme glouton ($k = 1$).



(c) Charge des liens en fonction de leur longueur pour les différents algorithmes étudiés.



(d) Espérance de la longueur du chemin calculé par les différents algorithmes étudiés.



(e) Taux des liens locaux sur les chemins calculés par les différents algorithmes étudiés en fonction de la distance de la source à la cible. *(échelle semi-log)*

Figure 1.8 – Paramètres statistiques de différents algorithmes de routage décentralisés (algorithme glouton et par exploration des contacts distants des ω bons contacts locaux) dans le cycle de Kleinberg $\mathcal{K}_{n,k=1}^{d=1}$. *(Extraits de la thèse d'Emmanuelle Lebhar [194, Chap. 3])*

- La figure 1.8(a) présente la probabilité qu'un nœud soit visité par l'algorithme glouton en fonction de sa position entre la source et la cible et du nombre k de contacts distants par nœud. Pour les petites valeurs de $k > 1$ (2 ou 3), on observe un comportement oscillant assez surprenant à proximité de la cible, où cette probabilité chute, remonte, puis rechute, avant de remonter. Ces oscillations s'expliquent intuitivement assez facilement car l'existence de plusieurs liens diminue fortement la probabilité de tomber immédiatement à côté du point de départ. On constate qu'en dehors de ce phénomène, la probabilité de visite d'un nœud varie assez peu lorsque k varie. Ces courbes sont aussi d'une très grande stabilité quand n augmente comme le démontre la figure 1.8(b) qui trace la position du point de probabilité de visite minimale pour $k = 1$. Celui-ci semble évoluer de façon parfaitement linéaire en n. En remarquant que lorsque l'on est loin de la cible, l'algorithme glouton avec k contacts distants est quasiment identique à l'algorithme par exploration locale avec $\omega = k$, il semble donc que ce paramètre ne permette pas de distinguer significativement l'algorithme glouton d'un algorithme disposant d'un peu plus d'informations.
- La figure 1.8(c) présente le paramètre le plus intéressant : l'espérance du nombre de chemins traversant une arête donnée en fonction de sa longueur. Ce paramètre, très facile à mesurer dans l'expérience de Milgram, présente en effet des distributions d'apparences assez différentes entre l'algorithme glouton et les algorithmes par exploration locale. Le premier semble en effet présenter une loi harmonique alors que les algorithmes par exploration locale donnent plutôt l'apparence d'une bosse très marquée de pente différente.
- Les figures 1.8(d) et 1.8(e) présentent enfin les espérances de deux caractéristiques des chemins calculés : leur longueur et le taux de liens locaux utilisés. Il apparaît clairement sur la figure 1.8(d) que l'algorithme par exploration locale n'est réellement plus efficace que l'algorithme glouton que lorsque la source est très loin de la cible, mais perd de son efficacité au fur et à mesure que l'on se rapproche de la cible. Ceci s'explique par la figure 1.8(e) qui démontre que le nombre de liens locaux empruntés par l'algorithme par exploration locale reste très élevé tout au long du chemin. Cet algorithme emprunte en effet $\omega/2$ liens locaux en moyenne à chaque phase d'exploration locale, alors que l'algorithme glouton emprunte beaucoup plus souvent des liens longs, ce qui devient très rentable à proximité de la cible.

Ces courbes ont révélé que la distribution de la charge des liens sépare nettement les différents algorithmes et pourrait donc être utilisée pour confronter le modèle de Kleinberg et ses variantes à l'expérience originale de Milgram [216]. Il serait également très intéressant d'obtenir des courbes similaires pour notre algorithme par explorations successives (non-locales) qui ne présente pas le même inconvénient que l'algorithme par exploration locale qui emprunte de trop nombreux liens locaux. Cependant, l'obtention d'équations exactes et manipulables pour cet algorithme semble inaccessible et il conviendrait donc d'adopter une autre approche, p. ex. basée sur des approximations garanties de ces paramètres. Je crois qu'une telle approche pourrait être réalisée en utilisant les mêmes techniques que nous avons développées dans la section précédentes et obtenir ainsi des estimations asymptotiques des lois de ces différents paramètres.

5 Émergence des petits mondes

Comme nous l'avons vu à la section 3, différents schémas de petit-mondisation ont été proposés récemment pour diverses classes de graphes [190, 293, 117, 3, 89]. Cependant aucun de ces schémas ne peut prétendre à expliquer comment ses structures apparaissent dans la nature, car ils reposent tous sur des constructions centralisées basées sur la connaissance globale du graphe initial. De tels processus ne sont pas envisageables dans le contexte d'un réseau social, où chaque individu ne connaît en moyenne qu'un millier de personnes environ. Aussi nous nous sommes intéressés à la faisabilité de ce modèle sous les contraintes des graphes réels, à savoir : peu de mémoire et un temps d'obtention très court, par rapport à la taille globale du réseau.

Avec Philippe Duchon, Nicolas Hanusse et Emmanuelle Lebhar [90], nous avons obtenu un algorithme totalement décentralisé ne consommant qu'une mémoire polylogarithmique par nœud, un temps polylogarithmique et n'échangeant qu'un nombre polylogarithmique de messages de taille polylogarithmique (polylogarithmique s'entend en la taille totale n du réseau).[22] Notre algorithme petit-mondise tout graphe à croissance α-bornée (pour $\alpha > 1$ constant) en ne rajoutant qu'un unique lien par nœud.

Cet algorithme pourrait avoir des applications intéressantes pour les réseaux pair-à-pair, puisqu'il permet d'en optimiser les performances (borne polylogarithmique sur la longueur du plus long chemin calculé par l'algorithme glouton entre deux sommets) par la simple application d'une phase d'optimisation de coût polylogarithmique en terme de temps, de mémoire et de messages échangés. Plusieurs protocoles ont été proposés par le passé pour maintenir des chemins courts entre toute paire de points d'un réseau pair-à-pair [297, 250, 326, 118]. Mais la plupart d'entre eux nécessite le maintient d'une structure sophistiquée alors que notre algorithme permet en une simple passe d'optimiser les performances du réseau sans maintenance : si les performances se dégradent, il suffit de le relancer pour retrouver un réseau performant.

Du fait de sa forme très algorithmique, notre processus n'a tel quel bien sûr aucun sens pour un réseau social. Cependant son existence démontre qu'il est possible de construire des graphes petit-monde avec très peu de ressource et en un temps très court à partir de graphes que l'on trouve dans la nature, ouvrant ainsi la possibilité de découvrir des processus plus naturels construisant de tels graphes. Comme nous le verrons en conclusion de cette section, cette voie est d'ailleurs en cours d'exploration avec le tout récent algorithme de Chaintreau, Fraigniaud et Lebhar [60] qui proposent pour la première fois une analyse rigoureuse d'un processus plausible conduisant à des petits-mondes, mais au prix d'une baisse significative d'efficacité (il nécessite un temps polynomial en la taille du réseau pour générer un petit monde).

5.1 Le modèle

Le graphe. Considérons un graphe H à croissance α-bornée et possédant n nœuds. Nous nous plaçons dans un modèle où chaque nœud possède une identité propre à partir de laquelle il est possible de calculer localement sa distance dans H à tout autre nœud. Rappelons qu'il existe des algorithmes classiques pour obtenir de telles identités, appelées étiquettes de distance, v. p. ex. [133, 4, 5]. Dans le cas d'une application aux réseaux pair-à-pair, nous pouvons simplement imaginer que les distances sont calculées de façon arbitraire à partir des adresses IP ou positions géographiques des nœuds, ou bien qu'elle est obtenue en mesurant

[22] J'ai essayé de caser un polylogarithmique de plus, mais je n'ai pas réussi... Ah si, tiens, dans la note en bas de page.

le temps moyen d'aller-retour d'un paquet entre les deux adresses IP des nœuds *(RTT, round trip time)* [326, 122].

Réseau virtuel et modèle des communications. Notre schéma repose sur la construction d'une structure encodant économiquement les tailles des boules à toutes les échelles dans le graphe. Cette structure consiste en un réseau virtuel (*overlay network*, en anglais), qui à l'instar d'Internet et à la différence d'un réseau physique, permet de communiquer directement avec tout nœud dont on connaît l'identité (l'adresse IP dans le cas d'Internet). Ainsi toute communication entre nœuds qui connaissent leurs identités réciproques, prend une unité de temps. Au départ, chaque nœud ne connaît que son identité propre et celles de ses voisins dans H. Nous considérons un modèle où chaque nœud peut recevoir ou émettre un unique message à chaque instant. Notre algorithme consiste en une alternance asynchrone de phases de calcul et de communications. Notre évaluation du temps sera basée sur le nombre maximum de phases de communication, appelées *round*.

5.2 Schéma totalement décentralisé de petit-mondisation

Notre schéma procède en deux phases :

- la construction d'un réseau virtuel encodant de façon économique une approximation à une constante près de la taille de chacune des boules du graphes ;
- la sélection des contacts distants de chaque nœud, tirés aléatoirement dans des listes obtenues via le réseau virtuel.

Construction du réseau virtuel. Dans un graphe à croissance α-bornée, connaître à une constante près les tailles des boules de rayons 2^i, pour $i = 1, \ldots, \lceil \log_2 n \rceil$, centrées sur des nœuds à distance au plus 2^i les uns des autres tels que ces boules couvrent le graphe, suffit pour estimer à une constante près la taille de toute boule de tout rayon centrée en tout point du graphe. Cette estimation nous suffira ensuite pour construire les liens longs d'une façon similaire à l'algorithme 1.2. Pour obtenir cette estimation de façon distribuée, nous allons contruire un réseau virtuel qui présente des similarités avec d'autres réseaux *overlay* classiques comme les *spanneurs* déformables [131] les *navigable nets* [158] ou les r*-nets* [158], dont le but principal est d'encoder une bonne approximation des distances dans H entre les nœuds. Notre objectif est différent : nous cherchons à obtenir une approximation de la taille des boules de rayons exponentiellement croissants, en particulier nous n'imposons pas de contrainte de distance minimale entre des nœuds d'un même niveau. Contrairement à ces autres structures, notre réseau virtuel encodant cette approximation pourra être obtenu efficacement de manière totalement distribuée.

Le principe de la construction de l'algorithme 1.6 est de sélectionner des nœuds à différents niveaux i allant de 1 à $\lceil \log_2 n \rceil$ tels que chaque boule de rayon 2^i contienne au moins un nœud de niveau i qui encodera sa taille. Pour ce faire, nous allons sélectionner itérativement des nœuds de niveau i croissant avec une probabilité *grosso modo* inversement proportionnelle à la taille de leur boule de rayon 2^i. Cette taille sera obtenue à partir des estimations des tailles des boules de rayon 2^{i-1} encodées dans les nœuds de niveau $i-1$. Afin de garantir un temps de calcul polylogarithmique, les nœuds de niveau i sont choisis parmi ceux de niveau immédiatement inférieur. Comme la construction de la structure au niveau i dépend des estimations des tailles des boules au niveau $i-1$, on pourrait craindre une propagation des

erreurs d'un niveau à l'autre. Pour l'éviter, notre structure est organisée de façon arborescente de sorte à couvrir totalement et sans redondance le graphe. En particulier, tous les sous-arbres sont *disjoints*. Lorsqu'un nœud passe du niveau i au niveau immédiatement supérieur $i+1$, il hérite des sous-arbres d'une partie des nœuds de niveau i à distance $O(2^i)$ dans H, qui ne sont pas passés au niveau supérieur. Comme nous ne pouvons pas garantir que son sous-arbre couvre correctement sa boule de rayon 2^{i+1}, il estime ensuite la taille de sa boule en se basant sur la taille des sous-arbres des nœuds de niveau $i+1$ proches de lui, à distance au plus $3 \cdot 2^i$ dans H. Les probabilités de sélection des nœuds sont ajustées pour qu'avec forte probabilité, chacun de ces calculs ne mette en jeu qu'un nombre polylogarithmique de nœuds, tous situés dans le voisinage immédiat des nœuds, et pour que ces estimations soient correctes avec forte probabilité.

Sélection des contacts distants. Le principe du schéma de petit-mondisation 1.2 est d'engendrer des liens uniformément à toutes les échelles de distance. Pour se faire, dans l'algorithme 1.7, nous demandons à chaque nœud de niveau i de tirer aléatoirement et uniformément un nombre logarithmique de nœuds dans l'approximation de sa boule de rayon 2^i. Cette approximation consiste en l'union de son sous-arbre et de ceux de ses voisins dans la structure arborescente construite à l'étape précédente. Les nœuds aléatoires sont calculés par descente récursive dans les sous-arbres en empruntant l'une ou l'autre des branches avec une probabilité proportionnelle à leurs tailles relatives (connues à l'issue de l'algorithme 1.6). Ensuite, chaque nœud sélectionne un niveau i aléatoire uniforme entre 1 et $\max_v \ell_v$ et prend pour contact distant un nœud aléatoire dans la liste de son ancêtre de niveau i. Ce procédé s'effectue de manière totalement décentralisée, en n'échangeant qu'un nombre polylogarithmique de messages par nœud. Il garantit de plus que les nœuds sélectionnés sont bien uniformément répartis sur les différentes échelles de distance. Ainsi, le graphe augmenté obtenu forme bien un petit-monde navigable, avec forte probabilité.

5.3 Construction décentralisée du réseau virtuel

L'algorithme 1.6 présente la construction du réseau *overlay* en détails. La structure arborescente du réseau virtuel est encodée de la façon suivante (v. illustration figure 1.9). Chaque nœud u de niveau au moins i mémorise :

- P_u^i, son *père* de niveau $i+1$. $P_u^i = u$ si u est de niveau au moins $i+1$.
- C_u^i, l'ensemble des nœuds de niveau $i-1$ (u inclus) dont u est le *père*.
- N_u^i, l'ensemble des nœuds de niveau au moins i à distance au plus $5 \cdot 2^i$ de u dans H, appelés les *voisins de niveau i de u*.
- $\dot{N}_u^i$, l'ensemble des nœuds de niveau au moins i à distance au plus $3 \cdot 2^i$ de u dans H, appelés les *voisins proches de niveau i de u*.
- ℓ_u, le niveau maximum atteint par u dans la structure.

D'autre part on note G_i le graphe ayant pour nœuds ceux de niveau au moins i et ayant pour arêtes toutes les paires de nœuds uv telles que $v \in N_u^i$ (ou de manière équivalente, $u \in N_v^i$). Seule une connaissance locale de ce graphe est nécessaire pour l'algorithme, mais il est pratique de pouvoir y faire référence dans l'analyse et la description de l'algorithme.

Les propriétés suivantes garantissent le bon fonctionnement de l'algorithme :

Algorithme 1.6 Construction décentralisée du réseau virtuel [90]

entrée(s) : Un graphe H à croissance α-bornée

1. **pour tout** nœud u **faire en parallèle**
2. Tirer $X_u \in [0,1]$ uniformément au hasard.
3. Poser $P_u^j := \perp$, $N_u^j := \varnothing$, $\dot{N}_u^j := \varnothing$ et $C_u^j = \varnothing$ pour $j \geqslant 1$.
4. Poser $T_u^0 := \{u\}$, $N_u^0 := \{v : \delta_H(u,v) \leqslant 5\}$, $\dot{N}_u^0 := \{v : \delta_H(u,v) \leqslant 3\}$, $P_u^0 := u$, $\ell_u := \perp$, $\beta_u^{-1} := 1$, $\beta_u^0 := b_u(1)$, et $i := 0$.
5. ▶ On notera G_j le graphe contenant les nœuds de niveau au moins j et ayant pour arêtes, les paires uv telles que $v \in N_u^j$ (ou de manière équivalente t.q. $u \in N_v^j$).
6. **tant que** $X_u < \frac{3\alpha^3 \log n}{\beta_u^i}$ et $\beta_u^i > \beta_u^{i-1}$ **faire**
7. Poser $i := i+1$. *// u passe au niveau supérieur.*
8. Poser $P_u^{i-1} := u$ et $C_u^i = \{u\}$. *// u est son propre père du niveau $i-1$.*
9. Envoyer le message "u a atteint le niveau i" à tous les nœuds à distance $\leqslant 3$ dans G_{i-1} (procéder par inondation de profondeur 3).
10. **traiter les messages en attente ou qui arrivent**
11. **à la réception d'un message** "v a atteint le niveau i" **faire**
12. **si** $\delta_H(u,v) \leqslant 5 \cdot 2^i$ **alors** ajouter v à l'ensemble N_u^i.
13. **si** $\delta_H(u,v) \leqslant 3 \cdot 2^i$ **alors** ajouter v à l'ensemble $\dot{N}_u^i$.
14. Accuser réception du message à v.
15. **à la réception d'un message** "v est ton fils" **faire**
16. Ajouter v à l'ensemble C_u^i.
17. **jusqu'à ce que** tous les nœuds à distance $\leqslant 3$ dans G_{i-1} aient répondu.
18. Notons T_u^i l'arbre enraciné en la copie de u de niveau i et ayant pour sous-arbres T_u^{i-1} et les T_v^{i-1} pour $v \in C_u^i$.
19. Demander aux nœuds $v \in \dot{N}_u^i$ la taille de leur sous-arbre T_v^i et poser :
20. $\beta_u^i := \sum_{v \in \dot{N}_u^i} \#T_v^i$, où $\#T_v^i$ désigne le nombre de feuilles de T_v^i.
21. *// Le niveau final de u est i.*
22. Poser $\ell_u := i$.
23. **si** $\beta_u^{\ell_u} = \beta_u^{\ell_u - 1}$ **alors**
24. *// Le réseau virtuel est terminé et u appartient au dernier niveau.*
25. Poser $P_u^{\ell_u} := u$ et envoyer à tous les nœuds de $(\cup_{j \geqslant 0} \dot{N}_u^j) \cup (\cup_{j \geqslant 1} C_u^j)$ le message "fin de l'algorithme".
26. **traiter les messages en attente ou qui arrivent**
27. **à la réception d'un message** "v a atteint le niveau j" **faire**
28. **si** $\ell_u = j-1$ et $P_u^{\ell_u} = \perp$ et $\delta_H(v,u) \leqslant 2^j$ **alors**
29. Poser $P_u^{\ell_u} := v$ et envoyer à v le message "u est ton fils".
30. **sinon**
31. Accuser réception du message à v.
32. **jusqu'à ce que** $P_u^{\ell_u} \neq \perp$ et que le message "fin de l'algorithme" soit reçu.
33. Envoyer le message "fin de l'algorithme" à tous les nœuds de $(\cup_{j \geqslant 0} \dot{N}_u^j) \cup (\cup_{j \geqslant 1} C_u^j)$.

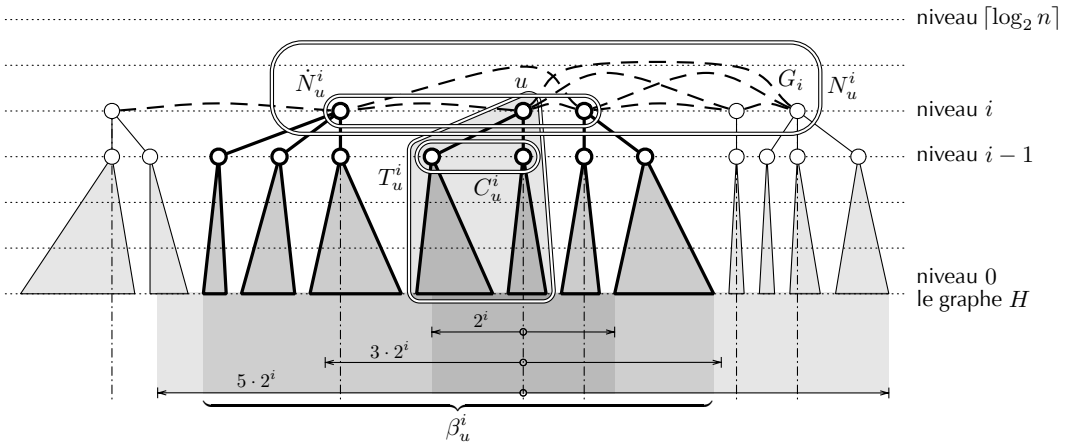


Figure 1.9 – Le réseau virtuel arborescent construit par l'algorithme 1.6. La forêt $\cup_{v \in \dot{N}_u^i} T_v^i$ enracinée sur les voisins proches de u au niveau i (les nœuds de niveau i à distance $\leqslant 3 \cdot 2^i$ de u dans H) couvre totalement les nœuds de la boule de rayon 2^i centrée sur u dans H et est totalement incluse dans la boule de rayon $5 \cdot 2^i$ centrée sur u dans H, avec forte probabilité. Le graphe G_i est représenté par des arêtes en pointillés entre les nœuds du niveau i à distance au plus $5 \cdot 2^i$ l'un de l'autre dans H.

- $\mathcal{P}_{\mathcal{C}}(i)$, *couverture :* Pour tout nœud $u \in H$, il existe un nœud v de niveau i dans la boule $B_u(2^i)$. Cette propriété garantit que le père de tout nœud de niveau $i-1$ est correctement défini à la ligne 29.
- $\mathcal{P}_{\mathcal{N}}(i)$, *voisinage :* Pour tout nœud u de niveau i, tous les nœuds de niveau i à distance au plus $5 \cdot 2^i$ de u dans H sont à distance au plus 3 de u dans G_i. Cette propriété garantit que les voisins de u dans G_i sont correctement calculés à la ligne 12.
- $\mathcal{P}_{\mathcal{A}}(i)$, *approximation :* $B_u(2^i) \subseteq \cup_{v \in \dot{N}_u^i} T_v^i \subseteq B_u(5 \cdot 2^i)$, pour tout nœud u de niveau i (v. illustration fig. 1.9). Cette propriété garantit que l'estimation des tailles de boules est correcte à une constante près.

Ces propriétés sont vérifiées avec forte probabilité (*i.e.*, avec probabilité $1 - O(\frac{1}{n})$) et s'obtiennent par récurrence sur les niveaux. La preuve repose sur les deux lemmes probabilistes suivants.

Lemme 1.23 ([90]). *Étant donné un rayon $r > 0$, soit S un ensemble de nœuds aléatoire obtenu en insérant chaque nœud u de H dans S indépendamment avec une probabilité p_u vérifiant :*

$$\min\left(1, \frac{C \log n}{b_u(r)}\right) \leqslant p_u \leqslant \frac{\gamma\, C \log n}{b_u(r)},$$

pour deux constantes $C \geqslant 3\alpha$ et $\gamma \geqslant 1$.

Alors, avec probabilité $1 - O(\frac{1}{n^2})$, pour tout $u \in H$, $B_u(r)$ contient au moins un nœud de S et $B_u(5r)$ contient au plus $2\alpha^4\, \gamma\, C \log n$ nœuds S.

Démonstration. Simple application des inégalités de Chernoff [225, section 4.1 page 67]. ■

Dans l'algorithme 1.6, les probabilités de passage au niveau suivant, ne sont pas indépendantes du processus de construction, mais le corollaire suivant va nous permettre de nous abstraire de ce problème de dépendance.

Corollaire 1.24 ([90]). *Soient $(X_u)_{u\in H}$ une suite de variables aléatoires indépendantes uniformes à valeurs dans $[0,1]$, un rayon $r>0$, et $(\beta_u)_{u\in H}$ une suite de* fonctions des (X_u) *vérifiant avec probabilité $1-O(\frac{1}{n^2})$, pour un certain $\gamma \geqslant 1$:*

$$b_u(r) \leqslant \beta_u \leqslant \gamma\, b_u(r), \quad \textit{pour tout } u \in H.$$

Soit S l'ensemble de nœuds aléatoire défini par :

$$S = \left\{ u : X_u < \frac{\gamma\, C \log n}{\beta_u} \right\}, \quad \textit{pour une certaine constante } C \geqslant 3\alpha.$$

Alors, avec probabilité $1-O(\frac{1}{n^2})$, pour tout $u \in H$, $B_u(r)$ contient au moins un nœud de S, et $B_u(5r)$ contient au plus $2\alpha^4\, \gamma\, C \log n$ nœuds de S.

Démonstration. Considérons les deux ensembles aléatoires suivants :

$$S_0 = \left\{ u : X_u < \frac{C \log n}{b_u(r)} \right\} \quad \text{et} \quad S_1 = \left\{ u : X_u < \frac{\gamma\, C \log n}{b_u(r)} \right\}.$$

Par hypothèse, les inclusions $S_0 \subseteq S \subseteq S_1$ sont vérifiées avec probabilité $1-O(\frac{1}{n^2})$. Ainsi, la borne inférieure du lemme 1.23 pour S_0 et la borne supérieure de ce même lemme pour S_1 implique les bornes inférieure et supérieure pour S avec probabilité $1-O(\frac{1}{n^2})$. ■

Ainsi, même si les valeurs des estimations des tailles des boules dépendent du processus probabiliste de construction, tant que nous pourrons garantir qu'elles sont correctes à un facteur constant près avec forte probabilité, nous pourrons nous affranchir de ces dépendances.

Prouvons à présent par induction que les propriétés $\mathcal{P}_{\mathcal{C}}(i)$, $\mathcal{P}_{\mathcal{N}}(i)$ et $\mathcal{P}_{\mathcal{A}}(i)$ sont vérifiées pour tout $0 \leqslant i \leqslant \lceil \log_2 n \rceil$, avec forte probabilité.

Lemme 1.25. *Pour tout $0 \leqslant i \leqslant \lceil \log_2 n \rceil$, si $\mathcal{P}_{\mathcal{C}}(j)$ et $\mathcal{P}_{\mathcal{N}}(j)$ sont vérifiées pour tout $j \leqslant i$, alors $\mathcal{P}_{\mathcal{A}}(i)$ est vraie.*

Démonstration. Considérons un nœud u de niveau i et $w \in B_u(2^i)$. Si $\ell_w \geqslant i-1$, par $\mathcal{P}_{\mathcal{C}}(i)$, le père $v = P_w^{i-1}$ de w est correctement défini et $d(v,w) \leqslant 2^i$. Sinon, par récurrence immédiate, w a un ancêtre v de niveau i tel que $d(v,w) \leqslant 2^i + 2^{i-1} + \cdots + 1 \leqslant 2^{i+1}$ par l'inégalité triangulaire. Dans tous les cas, $d(u,v) \leqslant 3 \cdot 2^i$, ce qui implique par $\mathcal{P}_{\mathcal{N}}(i)$ que v appartient à $\dot{N}_u^i$ et donc que $w \in \cup_{v\in\dot{N}_u^i} T_v^i$.

Considérons à present $v \in \dot{N}_u^i$. Par définition, $d(u,v) \leqslant 3 \cdot 2^i$. Comme la distance d'un fils à son père de niveau i est d'au plus 2^i, l'inégalité triangulaire donne que tous les nœuds de l'arbre T_v^i sont à distance au plus $2^i + 2^{i-1} + \cdots + 1 \leqslant 2^{i+1}$ de v. Il s'en suit que l'arbre T_v^i est totalement inclus dans la boule $B_u(5 \cdot 2^i)$. ■

Lemme 1.26. *Pour tout $0 \leqslant i \leqslant \lceil \log_2 n \rceil$, si $\mathcal{P}_{\mathcal{C}}(i-1)$ et $\mathcal{P}_{\mathcal{N}}(i-1)$ sont vraies, alors $\mathcal{P}_{\mathcal{N}}(i)$ aussi.*

Démonstration. Considérons deux nœuds de niveau i à distance $\delta \leqslant 5 \cdot 2^i$ l'un de l'autre dans H. Soit $u = w_0, \ldots, w_\delta = v$ une géodésique entre u et v dans H et posons $x = w_{\max\{0, \lfloor \delta/2 \rfloor - 2^{i-1}\}}$ et $y = w_{\min\{\delta, \lfloor \delta/2 \rfloor + 2^{i-1}\}}$. Par construction, $d_H(u,x) \leqslant 4 \cdot 2^{i-1}$, $d_H(y,v) \leqslant 4 \cdot 2^{i-1}$ et $d(x,y) \leqslant 2 \cdot 2^{i-1}$. D'après $\mathcal{P}_{\mathcal{C}}(i-1)$, ils existent deux nœuds s_1 et s_2 de niveau au moins $i-1$ appartenant à $B_x(2^{i-1})$ et $B_y(2^{i-1})$ respectivement. Ainsi, $d(u,s_1)$, $d(s_1,s_2)$ et $d(s_2,v)$ valent toutes au plus $5 \cdot 2^{i-1}$ et d'après $\mathcal{P}_{\mathcal{N}}(i-1)$, (u, s_1, s_2, v) forme un chemin de longueur 3 dans G_{i-1}, ce qui démontre $\mathcal{P}_{\mathcal{N}}(i)$. ■

Nous pouvons alors conclure la correction de l'algorithme.

Proposition 1.27. *Avec probabilité* $1 - O(\frac{\log D}{n^2})$, $\mathcal{P}_{\mathcal{C}}(i)$, $\mathcal{P}_{\mathcal{N}}(i)$ *et* $\mathcal{P}_{\mathcal{A}}(i)$ *sont vraies pour tout* $i \in \{0, \ldots, \lceil \log_2 n \rceil\}$, *où* D *désigne le diamètre du graphe* H.

Démonstration. Nous procédons par induction sur i. Clairement, $\mathcal{P}_{\mathcal{C}}(0)$, $\mathcal{P}_{\mathcal{N}}(0)$ et $\mathcal{P}_{\mathcal{A}}(0)$ sont vraies avec probabilité 1. Supposons que $\mathcal{P}_{\mathcal{C}}(j)$, $\mathcal{P}_{\mathcal{N}}(j)$ et $\mathcal{P}_{\mathcal{A}}(j)$ soient vraies pour $j < i$. D'après le lemme 1.26, $\mathcal{P}_{\mathcal{N}}(i)$ est vraie. Démontrons à présent $\mathcal{P}_{\mathcal{C}}(i)$.

D'après $\mathcal{P}_{\mathcal{A}}(i-1)$ et la croissance α-bornée de H, pour tout nœud u de niveau au moins $i-1$, nous avons

$$b_u(2^i) \leqslant \alpha\, b_u(2^{i-1}) \leqslant \alpha\, \beta_u^{i-1} \leqslant \alpha\, b_u(5 \cdot 2^{i-1}) \leqslant \alpha^3\, b_u(2^i).$$

Définissons donc β_u pour tout $u \in H$ comme suit :

$$\beta_u = \begin{cases} \alpha\, \beta_u^{i-1} & \text{si } u \text{ est de niveau au moins } i-1, \\ \max\{\alpha\, \beta_u^{\ell_u}, b_u(2^i)\} & \text{sinon.} \end{cases}$$

Par ce qui précède, nous avons : $b_u(2^i) \leqslant \beta_u \leqslant \alpha^3\, b_u(2^i)$. Considérons donc maintenant l'ensemble de nœuds aléatoire $S = \{u \in H : X_u < \frac{3\alpha^4 \log n}{\beta_u}\}$. Pour tous les nœuds de niveau au plus $i-1$, nous avons $\beta_u \geqslant \alpha\beta_u^{\ell_u}$ (v. ligne 6), et donc :

$$X_u \geqslant \frac{3\alpha^3 \log n}{\beta_u^{\ell_u}} \geqslant \frac{3\alpha^4 \log n}{\beta_u}.$$

Il s'en suit que S correspond exactement à l'ensemble des nœuds de niveau i, sélectionnés durant $i^{\text{ème}}$ itération de la boucle `tant que` de l'algorithme 1.6. L'application du corollaire 1.24 avec $C = 3\alpha$ et $\gamma = \alpha^3$ donne qu'avec probabilité $1 - O(\frac{1}{n^2})$, pour tout nœud u, la boule $B_u(2^i)$ contient un élément de S, c.-à-d. un nœud de niveau i. Ainsi $\mathcal{P}_{\mathcal{C}}(i)$ est vérifiée avec probabilité $1 - O(\frac{1}{n^2})$.

Enfin, sous l'hypothèse d'induction, puisque $\mathcal{P}_{\mathcal{C}}(j)$ et $\mathcal{P}_{\mathcal{N}}(j)$ sont vraies pour tout $j \leqslant i$ avec probabilité $1 - O(\frac{1}{n^2})$, $\mathcal{P}_{\mathcal{A}}(i)$ est vraie aussi avec la même probabilité, par le lemme 1.26.

Il s'en suit que la probabilité d'erreur sur l'ensemble des $\lceil \log_2 n \rceil$ itérations ne peut excéder $O(\frac{\log n}{n^2})$. ■

Proposition 1.28 (Complexité en temps, espace et communication). *Avec forte probabilité, le calcul du réseau virtuel par l'algorithme 1.6 nécessite une mémoire locale de* $O(\log^2 n \log D)$ *bits par nœud et s'effectue en moins de* $O(\log n \log D)$ *rounds de communications pendant lesquels* $O(\log n \log D)$ *messages de* $O(\log n)$ *bits sont envoyés et reçus par chaque nœud, où* D *désigne le diamètre du graphe* H.

Démonstration. Chaque message envoyé contient $O(\log n)$ bits (encodant l'identité d'un nœud). D'après le lemme 1.24, avec forte probabilité, tous les nœuds ont au plus $O(\log n)$ voisins ou enfants à mémoriser et à contacter durant chacun des $\lceil \log D \rceil + 1$ niveaux atteignables. ■

5.4 Schéma décentralisé d'augmentation

Commençons par remarquer que nous ne pouvons pas utiliser directement les liens père-fils de la structure arborescente du réseau virtuel pour construire les liens distants, car la connaissance locale des liens père-fils ne peut en aucun cas permettre de déterminer à quel sous-arbre appartient quel nœud. Aussi, aucun algorithme décentralisé (au sens de Kleinberg, définition 1.5 page 33) ne peut trouver son chemin dans cette structure localement, même avec la connaissance de la distance dans le graphe H.

La propriété importante des graphes à croissance α-modérée que nous allons utiliser ici, est leur isotropie : tirer $O(\log n)$ nœuds aléatoires uniformément dans une boule suffit à garantir que ces nœuds aléatoires en couvrent "toutes les directions", c.-à-d. couvrent toutes les boules de rayon moitié centrées dans la boule initiale. C'est la propriété qui va nous permettre d'obtenir un tirage économique et décentralisé de liens distants pertinents. L'algorithme 1.7 présente en détails le tirage décentralisé des liens distants.

Algorithme 1.7 Schéma décentralisé d'augmentation [90]

1) Échantillonnage des contacts distants potentiels :

pour toutes les copies des nœuds u aux différents niveaux atteints $i \in \{1, \ldots, \ell_u\}$ **faire en parallèle**

Sélectionner une liste $\mathcal{L}_u^i$ de $4\alpha^7\lceil \log n \rceil$ feuilles aléatoires choisies uniformément et indépendamment dans la forêt $\cup_{v \in N_u^i} T_v^i$ des sous-arbres enracinés dans le voisinage N_u^i de u au niveau i. (Procéder de façon distribuée en propageant $4\alpha^7\lceil \log n \rceil$ messages récursivement le long des arbres en les orientant aléatoirement et indépendamment sur les différents sous-arbres avec des probabilités proportionnelles aux tailles, connues, de ces sous-arbres, puis remonter les identités des feuilles atteintes directement à u).

Diffuser récursivement la liste $\mathcal{L}_u^i$ à tous les descendants de u dans T_u^i.

2) Détermination des contacts distants :

pour tous les nœuds u **faire en parallèle**

Tirer un niveau aléatoire uniforme $j \in \{1, \ldots, \max_v \ell_v\}$. ($\max_v \ell_v$ a été diffusé récursivement à tous depuis le sommet de la structure à la fin de l'algorithme 1.6)
Poser $L_u :=$ un nœud aléatoire uniforme de la liste $\mathcal{L}_s^j$, où s est l'ancêtre de u de niveau j ($s = u$ pour $j \leqslant \ell_u$).

Son principe est que chaque nœud constitue pour chacun des niveaux i qu'il a atteint, une liste $\mathcal{L}_u^i$ de $\Theta(\log n)$ nœuds aléatoires tirés uniformément dans la forêt $\cup_{v \in N_u^i} T_v^i$ approximant sa boule de rayon 2^i. Ces nœuds aléatoires sont obtenus par simple descente probabiliste le long des branches de la forêt jusqu'aux feuilles. Ensuite, chaque nœud tire un niveau i uniformément au hasard entre 1 et $\lceil \log_2 D \rceil$ puis sélectionne un nœud aléatoire dans la liste de son ancêtre de niveau i. Ce contact distant crée donc un lien de longueur $O(2^i)$ pointant vers une "direction aléatoire". De nouveau, ceci se fait en temps polylogarithmique de façon totalement décentralisée et garantit l'existence de liens à toutes les échelles et dans toutes les directions.

Pour démontrer que ces tirages engendrent bien un petit monde navigable, nous allons comme précédemment démontrer que l'algorithme glouton calcule avec forte probabilité des chemins de longueurs polylogarithmiques entre toutes les paires de nœuds du graphe augmenté engendré $G = (V, E, F)$, où $F = \{L_u : u \in V\}$.

Lemme 1.29 (Toutes les directions sont couvertes à toutes les échelles, [90]). *Pour tout niveau $i \geqslant 3$, pour tout nœud u de niveau au moins i, et pour tout nœud $v \in B_u(5 \cdot 2^{i-1})$,*

$$\Pr\{\mathcal{L}_u^i \cap B_v(2^{i-3}) \neq \varnothing\} \geqslant 1 - \frac{1}{n^4}.$$

Démonstration. La preuve repose sur une application de l'inégalité triangulaire, démontrant que la boule $B_v(2^{i-3})$ est couverte par la forêt $\cup_{w \in N_u^i} T_w^i$ dont elle représente une proportion constante (par croissance α-modérée), suivie de l'application de l'inégalité de Chernoff pour conclure qu'au moins un nœud de cette boule est sélectionné avec forte probabilité dans la liste de niveau i de u. ■

Corollaire 1.30 ([90]). *Avec probabilité $1 - O(\frac{\log D}{n^2})$, pour tout niveau $i \geqslant 3$, pour tout nœud u de niveau au moins i et pour tout $v \in B_u(5 \cdot 2^{i-1})$, la boule $B_v(2^{i-3})$ contient un nœud de $\mathcal{L}_u^i$.*

Il s'en suit que depuis tout nœud, la probabilité de diviser par deux la distance à une cible arbitraire est polylogarithmique.

Corollaire 1.31 ([90]). *Avec probabilité $1 - O(\frac{\log D}{n^2})$, pour toutes les paires de nœuds (u, v) telles que $2^{i-2} < d(u, v) \leqslant 2^{i-1}$, pour un certain $i \geqslant 3$, nous avons :*

$$\Pr\{L_u \in B_v(2^{i-3})\} \geqslant \frac{1}{4\alpha^7 \lceil \log n \rceil \lceil \log D \rceil}.$$

Nous pouvons alors conclure que le graphe obtenu est bien un petit monde navigable avec forte probabilité. Le théorème suivant présente une légère amélioration par rapport à notre article original [90], puisqu'il démontre que la borne sur l'espérance est en faite atteinte avec forte probabilité (gagnant ainsi un facteur $\log n$ sur la longueur du plus long chemin calculé par l'algorithme glouton par rapport à [90]).

Théorème 1.32 (petit-mondisation décentralisée). *Le graphe augmenté G engendré par les algorithmes totalement décentralisés 1.6 puis 1.7, est un petit monde navigable.*

Plus précisément, avec probabilité $1 - O(\frac{\log D}{n^2})$, l'algorithme glouton 1.1 (page 33) calcule des chemins de longueur au plus $O(\log^2 n \log D)$ entre toutes les paires de nœuds.

Démonstration. Il suffit de reprendre la preuve du théorème 1.10 (page 35) et l'utilisation de l'inégalité de Chernoff démontrant que pour toute paire de nœuds (u, v), avec probabilité $1 - O(\frac{1}{n^4})$, au plus $O(\log^2 n \log D)$ liens distants seront explorés avant d'atteindre v depuis u. Il suffit ensuite de sommer les probabilités d'erreur sur toutes les paires (u, v) pour obtenir le résultat désiré. ■

Remarquons que nous perdons un facteur $\log n$ sur la longueur du chemin calculé par l'algorithme glouton par rapport à l'algorithme de petit-mondisation centralisé de la section 3.1. Cette perte provient du fait que nous échantillonnons des sommets aléatoires dans les boules de façon uniforme et que pour couvrir toutes les boules de rayon moitié, nous sommes obligés d'échantillonner $\log n$ nœuds alors qu'un nombre constant (polynomial en α) suffirait. Il est probable que l'on puisse optimiser notre algorithme par un post-traitement de ces listes pour regagner ce facteur perdu.

6 Travaux postérieurs et perspectives

Ce chapitre a démontré, je l'espère, l'apport d'une approche algorithmique pour la compréhension d'un phénomène relevant typiquement des systèmes complexes. Les travaux initiaux de Kleinberg [188, 187] ont permis de saisir des rouages du phénomène petit-monde qui échappaient complétement à l'approche statistique proposée par Watts et Strogatz, par exemple. Kleinberg a d'ailleurs démontré que ces méthodes algorithmes pouvaient s'exporter à d'autres domaines, comme celui de l'évaluation de la pertinence des pages web avec son modèle des hubs et autorités [186]. La communauté mathématique lui a d'ailleurs décerné tout récemment le prix Nevanlinna pour la portée de ses travaux dans ces différents domaines [190].

Nous nous sommes attachés à approfondir les résultats de Kleinberg en explorant les limites du modèle, en proposant d'autres modèles pour les individus (algorithme par exploration locale) et des méthodes statistitiques (étude de la charge) pour aider à rechercher les algorithmes qui seraient utilisés inconsciemment dans la nature par les gens. Nous avons ensuite cherché à démontrer que ce modèle pouvait effectivement avoir une réalité en démontrant que l'on pouvait l'obtenir par un processus extrêmement efficace et économique en ressource puisque polylogarithmique en le nombre d'individus. Les deux algorithmes que nous avons proposés (algorithme de routage par exploration et algorithme décentralisé de petit-mondisation) pourraient d'ailleurs avoir d'intéressantes applications dans le domaine des réseaux pair-à-pair.

Une des questions centrales laissées ouvertes à la fin de la thèse d'Emmanuelle Lebhar était l'existence de graphes qui ne soient pas petit-mondisables. Fraigniaud, Lebhar et Lotker [120] y ont répondu par la négative en exhibant une classe de graphes de dimension doublante $\delta(n)$, qui ne peuvent être transformés en petit-mondes navigables par aucun schéma d'augmentation dès que $\delta(n) = \omega(\log\log n)$. Car alors, quelque soit le schéma d'augmentation, il existe toujours, une paire source-destination pour laquelle l'algorithme glouton calcule un chemin de longueur $\gg \operatorname{polylog} n$. Notons que ce résultat démontre en particulier l'optimalité des résultats de [89, 293].

D'autres résultats importants ont été obtenus depuis nos travaux. Deux approches me semblent particulièrement intéressantes. La première concerne l'émergence du phénomène petit monde avec l'algorithme *"move and forget"* de Chaintreau, Lebhar et Fraigniaud [60]. Ils ont démontré qu'associer une marche aléatoire à un processus d'oubli aléatoire suivant une loi harmonique, permet de petit-mondiser de façon uniforme toutes les grilles quelle qu'en soit la dimension. Ce processus, remarquablement simple et totalement décentralisé, est le premier analysé mathématiquement qui présente une explication plausible de l'émergence du phénomène petit-monde, à la nuance près qu'il requiert un temps de calcul irréaliste, polynomial en le nombre d'individus et ne saurait donc tout expliquer. Avant lui, un autre modèle plus efficace en temps avait été proposé, celui de Clauset et Moore [68] basé sur un processus de "ras-le-bol". Partant d'un anneau augmenté de façon arbitraire, il s'agit de lancer indéfiniment l'algorithme glouton sur des paires de sources et destinations aléatoires avec des "seuils de ras-le-bol" aléatoires. Si l'algorithme glouton n'atteint pas la cible avant que le nombre de pas n'excède le seuil, on force l'algorithme glouton à s'arrêter et l'on redirige le lien distant du dernier nœud atteint directement sur la cible. Clauset et Moore constatent expérimentalement que la distribution des longueurs des liens converge très rapidement vers la loi harmonique de Kleinberg. Cependant, malgré de nombreux essais menés par différents groupes, ce résultat n'a pas encore pu être démontré formellement. Une complication mathématique importante réside dans la redirection dynamique des liens. Ces deux résultats me semblent particulièrement intéressants et possèdent un potentiel d'amélioration certain.

La seconde approche qui m'a semblé novatrice est celle initiée par Chung, Lu, puis Anderson [67, 16] proposant une méthode pour retrouver dans les graphes réels les liens distants, dont l'existence est supposée par le modèle de Kleinberg. Leur idée est d'inférer ces liens par une évaluation de l'interconnectivité locale.[23] Ces méthodes furent ensuite adaptées et analysées par Fraigniaud, Lebhar, et Lotker [121] qui proposèrent un algorithme qui retrouve correctement les longs liens dans les modèles de grilles augmentées.

Ce domaine, à la frontière entre sociologie et optimisation des réseaux pair-à-pair, reste donc toujours très actif avec des productions originales, où l'algorithmique démontre qu'elle peut sans se travestir sortir de son cadre originel et proposer des éclairages pertinents sur les aspects calculatoires sous-jacents à de nombreux domaines extra-informatiques. Plus généralement, ce travail démontre l'existence d'une influence entre la structure d'un graphe et les fonctions qu'il remplit. Dans le cas du phénomène petit monde, des liens tirés suivant une loi liée harmoniquement à la métrique sous-jacente, permettent la fonction "router efficacement d'un point à un autre". Il serait très intéressant d'approfondir ce lien pour d'autres situations : on pourrait par exemple se poser la question de l'influence du moteur de recherche sur la structure du graphe du web et de l'action en retour de ces modifications sur son algorithme de classement des pages, et par exemple du risque que ce classement ne fonctionne plus, faute de liens pointant entre les pages (tout le monde préférant utiliser le moteur de recherche) ; on peut se poser également la question de l'influence globale des systèmes de recommandation sur les pages visitées d'un site, commercial ou bien entre blogs ; ou bien encore, l'influence de la structure des villes sur la mixité socio-profesionnelle ; ou enfin, l'influence de telle structure de graphe dans un réseau de gènes par rapport à la fonction qu'il est sensé remplir. Il est tout à fait possible qu'une approche algorithmique puisse apporter un point de vue complémentaire révélant des relations insoupçonnées.

[23] *Clustering coefficient*, en anglais.

2 Régimes transitoires dans les réseaux d'automates

Les travaux présentés dans ce chapitre trouvent leur origine dans la partie expérimentale de la thèse de Nazim Fatès, dirigée par Michel Morvan entre 2002 et 2005. Avec Éric Thierry, nous étions convaincus qu'une grande partie des constatations expérimentales de Nazim Fatès pourrait être prouvée mathématiquement, ce que nous avons commencé à faire dès 2004 avec lui. Nous avons ensuite co-encadré le stage de M2 puis la thèse de Damien Regnault entre 2005 et 2008. Avec Damien, nous avons pu approfondir et étendre nos premiers résultats, en proposant en particulier un début d'analyse des transitions de phases étudiées expérimentalement par Nazim Fatès, puis une extension de nos études à la dimension 2 qui a révélé des structures combinatoires sophistiquées et assez inattendues. Les travaux présentés ici ont été publiés dans [110, 111, 112, 255, 256, 258, 257] et dans [251, 253, 252] pour ceux réalisés par Damien Regnault seul durant sa thèse.

1 Introduction

Contexte de notre étude. Depuis la découverte de la structure de l'ADN en 1953 [314], le séquençage de l'ADN laissait entrevoir à certains la possibilité de lire dans le vivant comme dans un texte. Ces espoirs se sont malheureusement rapidement heurtés à l'intrication du réseau de régulation génétique et à la complexité qui en résulte.[24] Bien que certains organismes soient totalement séquencés, et qu'ils ne possèdent qu'une quarantaine de gènes seulement et que leurs réseaux de gènes soient totalement connus [71], personne n'est actuellement capable de décrire ni de prédire le déroulement de la croissance des organismes résultants.

Ces études ne portent pourtant que sur des organismes isolés, laissant de côté les nombreuses interactions symbiotiques entre les différentes espèces. Le talk de Bassler [34] à TED [300] est particulièrement instructif sur le niveau très élevé des intrications interspécifiques. Après avoir rappelé que nous, êtres humains, sommes constitués de bactéries indispensables à notre survie à plus de 90% en terme de nombre de cellules (env. 10^{12} cellules humaines pour 10^{13} cellules bactéries sur un être humain) et à plus de 99% en terme de matériel génétique (30 000 gènes humains pour 3 000 000 gènes bactériens), elle évoque le cas étonnant d'une

[24] Une complexité qui était évidente pour toute personne ayant une culture minimale en science informatique.

pieuvre contenant en son sein des bactéries luminescentes qui lui servent la nuit à annuler son ombre durant ses périodes de chasses. Au petit matin, elle ouvre la poche qui les contient pour en éliminer la plus grande grande partie et en limiter ainsi la croissance, mais en conserve suffisamment pour s'assurer qu'elles seront de nouveau en quantité suffisante pour émettre cette indispensable lumière le lendemain soir. On y apprend également que les bactéries communiquent entre elles par émissions de molécules spécifiques qui lorsqu'elles atteignent un seuil suffisant déclenchent une même action simultanée de toutes les bactéries présentes. Ce processus leur permet par exemple de lancer une attaque (p. ex., une infection) lorsqu'elles sont en nombre suffisant pour supporter la réponse de l'organisme parasité. Ce système présente de très grandes similarités avec le modèle des protocoles de populations étudié par exemple dans [54]. Dans cet article, les auteurs étudient la puissance de calcul d'un tel système où les agents ont un nombre fini d'états et changent d'état en fonction de leurs rencontres aléatoires avec d'autres agents suivant des règles de type automates cellulaires : *e.g.*, si les états des deux agents qui se rencontrent sont Bleu et Rouge, alors ils passent dans les états Vert et Jaune respectivement. Dans le cas des bactéries, ces rencontres aléatoires uniformes sont obtenues par la propagation des molécules-messages (l'état de la bactérie en quelque sorte) émises dans le milieu qui vont se fixer sur les récepteurs d'autres bactéries, aléatoirement par agitation thermique. Ainsi ces travaux portant sur un modèle d'apparence extrêmement abstraite pourraient avoir d'intéressantes applications dans le domaine médical.

Une caractéristique singulière des réseaux de gènes, que nous avons évoquée dans l'introduction générale de ce manuscrit, est la relative lenteur de leurs transitions : environ 10 à 12 minutes pour qu'un gène commute entre les états activé et inhibé ; environ 10 à 13 minutes pour qu'une "molécule codant de l'information" passe d'une cellule à une autre (v. les cours de Prochiantz au collège de France [242]). Il s'agit donc de n'envisager que quelques centaines de transitions par jour ! Il semble donc illusoire de se focaliser sur les distributions limites de ces systèmes.[25]

Dans l'exemple de la morphogenèse, il semble aussi très difficile de faire abstraction de la géométrie comme pour les bactéries. Par exemple, durant la croissance de l'organisme, la multiplication des cellules est un facteur au moins aussi important que les transitions d'états dans les réseaux de gènes de chacune de ces cellules. La multiplication cellulaire permet d'ailleurs probablement de compenser la relative lenteur des transitions. La figure 2.1 pré-

Figure 2.1 – Les motifs de la coquille du *conus textile* (à droite) croissent suivant une règle mathématique assez similaire à celle de l'automate cellulaire élémentaire **30** (à gauche).

[25] La distribution limite d'un organisme vivant n'est-elle d'ailleurs pas la mort ?

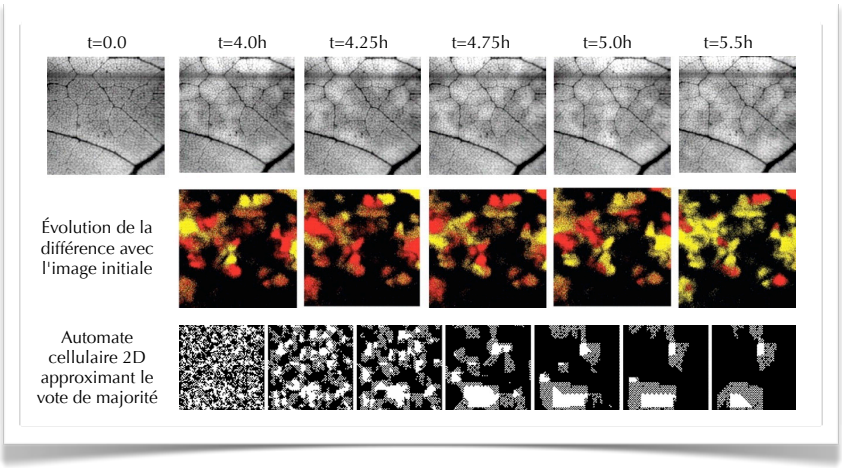


Figure 2.2 – Sur les deux lignes du haut : Évolution des motifs de l'activité des stomata (pores) d'une feuille de *Xanthium strumarium* à la recherche d'un nouvel équilibre dans un environnement constant et uniforme après une chute brutale de l'humidité ambiante à $t = 0$. Les auteurs de [236, 223] y démontrent par des arguments statistiques que les corrélations spatiales et temporelles sont similaires à celles d'automates cellulaires 2D tels celui approximant le vote de majorité, représenté en dessous. *[Images extraites de l'article [223]]*

sente l'exemple d'un coquillage dont les motifs croissent suivant une règle mathématique très similaire à celle gouvernant des automates cellulaires bien connus. Du fait de la relative lenteur de ces systèmes plongés dans des espaces métriques, il semble très improbable que leurs transitions aient lieu de manière synchrone.

Peak, West et Mott [236, 223] ont étudié p. ex. le cas de certains végétaux qui ne présentent pas a priori de contrôle central : contrairement aux animaux, ils ne possèdent ni système nerveux ni hormonal. Les pores *(stomata)* des feuilles de ces végétaux doivent résoudre le problème suivant : s'ouvrir pour permettre à la plante de s'alimenter en CO_2 mais pas trop pour éviter que la plante ne se déshydrate. Lorsque le milieu est stable, l'activité des pores de la feuille est uniforme (v. illustration fig. 2.2 à $t = 0$). En revanche dans le cas d'une brusque chute de l'humidité ambiante, les auteurs observent l'apparition de motifs sur l'activité des pores (observée via la photosynthèse engendrée par la conversion du CO_2), et ce bien que le nouveau taux d'humidité soit uniforme sur toute la surface de la feuille. Ils évaluent alors les corrélations spatiales, spectrales et temporelles de ces motifs et démontrent par des arguments statistiques (analyse de Fourier et densité de probabilité) que l'évolution des motifs présente les mêmes caractéristiques statistiques que celles des motifs mesurés pour certaines règles d'automates cellulaires. L'hypothèse explorée par les auteurs est que les pores de la feuille cherchent à établir un consensus avec leurs voisins sur leur niveau d'ouverture. Ils comparent donc les motifs observés avec ceux d'automates cellulaires connus pour approximer le consensus majoritaire, c.-à-d. celui où toutes les cellules finissent par passer dans l'état majoritaire sur l'ensemble de la configuration initialement, v. [75] (v. illustration fig. 2.2). Leurs calculs révèlent alors une très bonne concordance statistique. Ils en concluent que les phénomènes observés sont statistiquement équivalents à ceux observés dans les automates cellulaires. Ces résultats démontrent selon eux en particulier l'inadéquation des approches continues de type champ moyen (à base d'équations différentielles) qui ne présentent pas,

elles, ces motifs hétérogènes et présupposeraient donc l'existence d'un contrôle global, absent dans de tels organismes.

Une autre caractéristique de certains systèmes vivants qui plaide en faveur d'une modélisation par automates est la surprenante indépendance de leur évolution par rapport aux valeurs des concentrations des différents intervenants. En effet, des expériences mentionnées par Prochiantz dans son cours au Collège de France [242] démontrent qu'à certaines étapes clé du développement d'un organisme, si l'on efface chimiquement un gène dans une cellule où il devrait être exprimé, la cellule parvient à le réexprimer à la bonne concentration au bout de quelques heures et à poursuivre son développement normal. Le fait que ce gène doive être exprimé à cette concentration à ce moment-là de l'évolution, est donc encodé quelque part dans l'*état* de la cellule, sans doute par des mécanismes de redondance du réseau de régulation génétique, ce qui témoigne une nouvelle fois de son étonnante complexité. La valeur précise de cette concentration, bien qu'indispensable à la poursuite de l'évolution, n'est donc pas capitale et peut être corrigée par la suite par la cellule elle-même, dont *"l'état"* mémorise en quelque sorte cette valeur. On peut donc espérer qu'une modélisation puisse faire abstraction de ces concentrations et se focaliser sur des règles d'évolution discrètes entre des états différents de la cellule.

Demongeot et coll. proposent effectivement dans [79] de modéliser les réseaux génétiques sous la forme de réseaux booléens. Ils y démontrent en particulier que le nombre d'attracteurs est relié au nombre de cycles positifs[26] et observent expérimentalement que cette relation mathématique est cohérente avec le nombre de différentiations possibles des cellules de plusieurs organismes. Dans [19], les auteurs généralisent leur approche par réseau booléen en adjoignant le réseau métabolique au réseau de régulation génétique dans le cas de la bactérie *E. Coli*. Ce modèle leur permet de rendre compte correctement des différents modes d'alimentation de la bactérie : chacun de ces modes correspond à un des points fixes du réseau modélisé. Ils ont cependant été contraints d'introduire trois composants "jokers" pour que le réseau ait le comportement souhaité. Ils en déduisent que cette méthode mathématique permet de détecter lorsque les graphes de régulation ou de métabolisme connus sont incomplets, et qu'elle peut donc permettre de découvrir l'existence de nouvelles voies métaboliques encore inconnues expérimentalement. La validité de l'approche par réseaux booléens des mécanismes de régulations est confortée par d'autres travaux récents, v. p. ex. [81, 80].

Notons que des systèmes de ce type sont également utilisés en sociologie, p. ex. pour modéliser la ségrégation sociale [284, 285].

Notre approche. Les résultats étonnants obtenus depuis les années 1970 via l'injection d'aléatoire dans les algorithmes ont démontré combien notre intuition peut être fausse sur les ressources nécessaires pour poursuivre un calcul, et plus généralement produire une dynamique (v. l'introduction générale de ce manuscrit). Tout particulièrement, les compteurs log log de Flajolet et coll. [115] ont démontré que pour compter jusqu'à n, $\log\log n$ bits suffisent de façon probabiliste alors qu'on serait tenté de penser que $\Omega(\log n)$ sont nécessaires. Le gain est énorme puisqu'exponentiel. Si l'on imagine que l'on recherche au hasard un algorithme de comptage, nous aurons trouvé l'algorithme de comptage aléatoire, avant d'avoir fini d'écrire le compteur déterministe équivalent ! (et donc en utilisant exponentiellement moins de ressources) Il me semble donc tout à fait légitime de penser que ces processus aléatoires exponentiellement plus économes ont infiniment plus de chances d'apparaître dans la nature que les processus déterministes, plus sophistiqués et exponentiellement plus coûteux en res-

[26] Un cycle est positif s'il contient un nombre pair d'actions inhibantes.

sources.

Il nous a donc semblé intéressant d'explorer de manière frontale, c.-à-d., sans distinction a priori, *l'ensemble* des comportements possibles de tels systèmes. Nous nous sommes focalisés sur les systèmes de petite taille et ce pour deux raisons. D'une part, nous pensons qu'entre deux systèmes qui font la même chose, le plus petit a plus de chance de surgir dans la nature (car il a plus de chance d'apparaître par hasard). D'autre part, car ils se laisseront plus facilement analyser. Notons enfin que l'étude de petits systèmes sera de toute façon une étape nécessaire à franchir avant d'analyser des systèmes plus gros. Notre but est à la fois de répertorier les différents comportements possibles et de proposer des méthodes d'analyse génériques et performantes. Nous avons donc cherché à développer des méthodes d'écriture de preuves qui soient à la fois compactes et faciles à relire. En effet, un des obstacles majeurs auxquels nous avons dû faire face dans nos analyses sont les *énumérations de cas*, dont il faut être certain qu'elles sont exhaustives et cohérentes, *i.e.* que l'on puisse certifier aisément que les fonctionelles que nous définissons à partir de ces énumérations sont correctement définies et vérifient les propriétés souhaitées.

Loin d'être une question anecdotique, je suis convaincu que réussir à effectuer des énumérations de cas de manière efficace mathématiquement sera un des pas décisifs vers la constitution d'une science des systèmes complexes. Ceci la distinguera singulièrement des autres sciences (mathématique et physique) dont l'un des objectifs est justement de trouver des voies de contournement pour éviter ces énumérations (recours au continu, méthode du champ moyen, etc.). Il s'agit de plus d'une direction de recherche naturelle pour les informaticiens qui ont développé depuis la création de l'informatique une intuition profonde du discret et du formalisme de la preuve mathématique. Ces énumérations sont de plus extrêmement instructives et nous verrons que les méthodes que nous avons développées, permettent d'aller beaucoup plus loin.

Autres travaux portant sur des modèles proches. De nombreux articles ont été consacrés à différents processus de type réseaux d'automates. La plupart porte sur l'étude de processus particuliers. Parmi les processus qui ont fait l'objet d'études particulièrement approfondies, on peut citer les modèles d'Ising [237, 232] et de Potts [327], la percolation dirigée ou non [152], les réseaux d'Hopfield [165], les modèles de vote et processus d'exclusion [12, 199, 200]. Une grande partie de ces travaux est consacrée à l'étude des distributions limites, et de leur existence en particulier. La plupart de ces modèles, regroupés sous le terme particules en interaction [199, 200], sont issus de la physique statistique et présentent donc une dynamique en perpétuelle évolution, le plus souvent sur des configurations infinies, où même les transitions les plus improbables ont toujours une probabilité positive d'arriver.

Les dynamiques que nous allons considérer dans ce chapitre sont dans un certain sens plus déterministes, puisque seules les transitions compatibles avec la règle de l'automate pourront être exécutées. Si les modélisations de type thermodynamique s'appliquent naturellement aux gaz, il nous semble assez douteux qu'elles s'appliquent aux processus biologiques. Les tâches d'un léopard ne se déplacent pas aléatoirement *ad vitam æternam*, mais sont construites par un processus *fini dans le temps* qui, certes, s'alimente sans doute d'aléatoire, mais qui est avant tout gouverné à la manière des algorithmes randomisés par des règles déterministes dictées (entre autres) par le "code génétique" que l'on modélise souvent par des automates (v. p. ex. [79, 80, 81]).

Comme nous l'avons vu dans l'introduction de ce manuscrit, les transitions de ces systèmes sont de plus très lentes par rapport à la durée de vie des organismes hôtes (env. 10 minutes

par transition, soit moins d'un million de transitions seulement jusqu'à la fin de l'adolescence pour un être humain ! à comparer aux env. 10^{14} cellules nous constituant). Nous avons donc choisi de nous concentrer sur l'étude des étapes transitoires de ces processus.

Automates cellulaires asynchrones. Partant de la constatation que pour au moins une bonne part des mécanismes en jeu en biologie, les cellules évoluent de façon asynchrone, nous nous sommes interrogés sur l'influence que pouvait avoir l'asynchronisme sur le calcul des automates cellulaires. Plusieurs études expérimentales (v. p. ex. [58, 45, 286]) avaient démontré auparavant que l'introduction d'asynchronisme dans les mises-à-jour produit des changements radicaux sur l'évolution des automates cellulaires. En régime asynchrone, les configurations ont alors tendance à converger rapidement vers des configurations stables, alors que les évolutions synchrones produisent typiquement des diagrammes espace-temps d'un des quatre types suivants (v. [323, 325]) : *nilpotent* (convergent rapidement vers une configuration stable), *cyclique* (convergent vers un cycle fini de configurations), *"chaotique"* (produisant des images bruitées sans structure apparente) ou *"complexe"* (produisant des motifs manifestement structurés où des signaux, ressemblant à des paquets bruités, se déplacent, entrent en collision, donnent naissance à d'autres signaux, et semblent conduire ainsi des calculs).[27] Les simulations conduites semblent indiquer que cette classification s'écroule en régime asynchrone où la plupart des automates adoptent des comportements de type nilpotent ou "chaotique léger" (c.-à-d., faiblement bruités). Nous verrons que certains de ces nilpotents adoptent cependant des comportements de type "complexe" (p. ex., des particules aux déplacements sophistiqués) durant leur phase transitoire.

Travaux reliés sur les automates probabilistes. Une partie de la communauté mathématique probabiliste s'intéresse aux automates cellulaires probabilistes en tant que tels, et plus particulièrement à leur comportement limite pour $t \to \infty$. Un automate probabiliste est un automate où, à chaque pas de temps, chaque cellule choisit indépendamment la règle selon laquelle elle va se mettre à jour suivant une distribution partagée par toutes les cellules. Les automates asynchrones sont le cas particulier où chaque cellule a le choix entre la règle identité et la règle de l'automate considéré. Un sujet de préoccupation centrale de ce domaine est de définir des conditions pour l'unicité ou non du comportement à l'infini, et en particulier sur l'existence d'une unique mesure stationnaire. Ce problème est intimement lié au problème de la résistance aux pannes en informatique où il s'agit de faire en sorte que le comportement reste asymptotiquement le même quelques soient les perturbations imposées au système. Prouver l'existence de plusieurs mesures invariantes en présence de bruit (p. ex., engendré par un adversaire malveillant autorisé à modifier chaque cellule avec probabilité $\varepsilon > 0$ pour ε suffisamment petit), permet d'assurer que cet automate permettra de mémoriser la valeur d'au moins un bit. Toom a démontré que c'était possible avec un automate bidimensionnel à deux états [304]. Son travail a été par la suite simplifié par [44] puis repris par Gács et Reif en 1988 [130] pour obtenir un automate cellulaire 3D capable de mener des calculs en présence d'un adversaire malveillant ayant le pouvoir de modifier indépendamment chaque cellule avec probabilité $\varepsilon > 0$ suffisamment petit. Dans [128, 129], Gács étendit alors son résultat au cas d'un automate cellulaire 2D puis 1D, proposant alors le premier automate

[27] Cette classification expérimentale des automates cellulaires synchrones, proposée par Wolfram [323, 325], a été à l'origine de développements théoriques très intéressants sur la classification des comportements des automates déterministes (v. p. ex. les travaux de l'équipe de Jacques Mazoyer [249, 230, 302] et l'état de l'art [231]). La quatrième classe, les "complexes", est sans nul doute la plus intéressante car c'est là qu'on y trouve les automates les plus puissants. Notons que la notion de "signaux" et de sa contrepartie, la notion de "motifs de fond", n'ont toujours pas pu être définies de façon satisfaisante.

unidimensionel capable de mener des calculs fiables en présence d'erreurs se produisant sur chaque cellule avec probabilité $\varepsilon > 0$. Ce dernier résultat est particulièrement intéressant car il contredit une conjecture qui était largement admise, selon laquelle : tout automate unidimensionnel où chaque cellule peut, par une mise-à-jour, toujours atteindre n'importe quel état avec probabilité positive, admettrait une unique mesure invariante. Ce résultat reste cependant très controversé car sa preuve est particulièrement difficile (plus de 200 pages) ; dans [150], Gray estime à environ 2^{32} le nombre d'états de l'automate construit par Gács et à 2^{-960} le taux d'erreurs qu'il supporterait.

Ces études consistent donc principalement à concevoir des automates extrêmement complexes pour (in)valider des conjectures plus générales. Nous adoptons dans ce chapitre une approche radicalement différente en apparence : l'exploration exhaustive des comportements possibles des petits automates sans a priori. Nos méthodes sont aussi très différentes, puisque nous nous intéresserons à l'espérance du temps de convergence des automates et non à leur mesure limite. Pourtant, nous verrons en conclusion de ce chapitre que les questions que nous nous posons et les questions posées par le domaine des automates probabilistes pourraient bien être les deux versants d'une même montagne. Nous avons en effet exhibé un certain nombre d'automates uni- et bi-dimensionnels présentant des comportements suffisamment complexes pour qu'on puisse espérer pouvoir faire un lien entre la "polynomialité" ou non de leurs temps de convergence et l'existence ou non de plusieurs mesures invariantes.

Notre contribution. Avec Nazim Fatès, Michel Morvan et Éric Thierry, puis Damien Regnault, nous nous sommes tout d'abord intéressés aux automates cellulaires élémentaires (1D à deux états). Le travail de Nazim Fatès porte sur l'étude de la robustesse à l'asynchronisme, c.-à-d. sur les modifications macroscopiques induites par des mise-à-jour asynchrones des automates. Son étude comportait initialement une forte composante expérimentale révélant que dans certains cas les configurations convergeaient très vite vers des états stables, alors que dans d'autres, elles semblaient continuer d'évoluer à jamais sans progrès apparemment. Le problème qui se posait alors était de déterminer si l'on pouvait tirer les bonnes conclusions des diagrammes espace-temps observés, autrement dit si le temps de simulation était suffisamment long pour rendre compte de tous les phénomènes possibles. Nous avons donc démarré avec Éric Thierry une étude théorique de ces automates et démontré qu'en régime totalement asynchrone où une unique cellule aléatoire est mise à jour à chaque pas de temps, à l'exception d'un seul automate (à symétrie près), ceux qui convergent, convergent en temps polynomial et leur convergence peut donc bien être observée. Notre étude est basée sur l'étude d'un paramètre algorithmique naturel : le *temps de relaxation*, c.-à-d. l'espérance du temps de convergence depuis la pire des configurations initiales. La conclusion de notre travail a été que non seulement l'étude de ce paramètre permet de valider les études expérimentales, mais aussi de *classifier correctement* les différents automates que nous avons étudiés. En effet, les diagrammes espace-temps des automates ayant le même temps de relaxation ont les mêmes structures, gouvernées par des processus aléatoires sous-jacents identiques. Nous avons ensuite généralisé notre étude avec la thèse de Damien Regnault au cas des régimes α-asynchrones où à chaque pas de temps, chaque cellule se met à jour ou pas indépendamment avec probabilité α. De nouveaux phénomènes apparaissent qui compliquent considérablement l'analyse, mais nous démontrons que le temps de relaxation permet toujours de caractériser correctement le comportement de chacun des automates et d'y associer un processus probabiliste représentant de sa catégorie. Les nombreuses énumérations de cas imposées par la complexité de certaines situations nous ont contraint à introduire un nouveau formalisme,

les *arbres de masques* et *arbres d'analyse*, dont nous pensons qu'ils pourraient s'avérer très utiles pour simplifier la rédaction des preuves sur les automates cellulaires probabilistes souvent fastidieuses et peu lisibles. Sous l'impulsion de Cris Moore, nous avons ensuite cherché à généraliser notre approche aux automates bi-dimensionnels, beaucoup plus nombreux. Notre choix s'est tout d'abord porté sur la règle de minorité pour les voisinages des quatre et des huit plus proches voisins pour des raisons que nous développerons à la section qui lui est consacrée. Nous avons été très surpris par l'étonnante richesse comportementale de cet automate dont nous n'avons pu analyser que partiellement le régime totalement asynchrone. En conclusion de ces différentes études, nous isolons quelques automates (1D et 2D) aux comportements plus compliqués, pour lesquels nous soupçonnons que les transitions de phase observées sur leur temps de relaxation soient reliées intimement aux transitions de phase conjecturées sur le nombre de leurs mesures invariantes par la communauté mathématique.

Après une section introduisant les notations utilisées (qui peut être lue très rapidement), nous présentons à la section 3 nos résultats expérimentaux puis théoriques de l'analyse des différents régimes asynchrones des automates cellulaires unidimensionnels. La section 4 présente quant à elle les généralisations de nos analyses aux automates bidimensionnels. Ce chapitre se termine par des conjectures et des problèmes ouverts, incluant les connexions sur lesquelles nous travaillons avec la communauté mathématique des automates probabilistes.

2 Automates cellulaires et dynamiques asynchrones

2.1 Automates cellulaires

Définition 2.1 (Configuration). *Étant donné un ensemble de* cellules $(\mathbb{T}, +)$ *muni d'une structure de groupe (typiquement* $(\mathbb{Z}_n^d, +)$*, ici) et un ensemble fini d'*états S*, on appelle* configuration *c, toute fonction $c : \mathbb{T} \to S$. On dit que la cellule i est dans l'état c_i. On note $N = \#\mathbb{T}$, la taille de la configuration, c.-à-d. le nombre total de cellules.*

Définition 2.2 (Automate cellulaire). *Un* automate cellulaire *sur $(\mathbb{T}, +)$ est défini par :*

- *un Δ-uplet $V = (\partial_1, \dots, \partial_\Delta) \in \mathbb{T}^\Delta$, appelé* voisinage *et définissant un* réseau *sur les cellules : les voisins d'une cellule i sont les cellules $j_1, \dots, j_\Delta$ avec $j_k = i + \partial_k$ pour $k = 1, \dots, \Delta$ (i est son propre voisin si $\exists k, \partial_k = 0$).*
- *une* fonction de transition *$\delta : S^\Delta \to S$, qui donne le nouvel état d'une cellule i en fonction des Δ états de ses voisines lorsque celle-ci est mise à jour.*

Un automate cellulaire sera désigné par le quadruplet $((\mathbb{T}, +), S, V, \delta)$. Le couple cellules-relation de voisinage, $((\mathbb{T}, +), V)$, est appelée la topologie *de l'automate. Le Δ-uplet des états des cellules voisines d'une cellule i, $(c_{i+\partial_1}, \dots, c_{i+\partial_\Delta})$, est appelé le* voisinage *de i dans la configuration c.*

À topologie fixée, il y a donc $(\#S)^\Delta$ voisinages possibles et donc $(\#S)^{(\#S)^\Delta}$ fonctions de transitions différentes, et donc autant d'automates cellulaires différents.

2.2 Régimes synchrone et asynchrones

Différentes politiques de mise-à-jour peuvent être envisagées pour un automate cellulaire.

Définition 2.3 (Dynamique). *On appelle* dynamique $(D_t)_{t\geqslant 0}$ *d'un automate cellulaire* $(\mathbb{T}, S, V, \delta)$*, toute suite de* variables aléatoires *à valeurs dans les ensembles de cellules,* $D_t \subseteq \mathbb{T}$*. Nous dirons que les cellules* $i \in D_t$ *sont celles qui se* mettent à jour *au temps* t*.*

Partant d'une configuration initiale c^0 *au temps* $t = 0$*, les configurations* c^t *aux dates ultérieures* $t \geqslant 1$ *sous la dynamique* (D_t) *sont données par la relation récursive suivante :*

$$\text{pour } t \geqslant 1, \text{ pour tout } i \in \mathbb{T}, \quad c_i^t = \begin{cases} \delta(c_{i+\partial_1}^{t-1}, \dots, c_{i+\partial_\Delta}^{t-1}) & \text{si } i \in D^t \\ c_i^{t-1} & \text{sinon} \end{cases}$$

Notons que $(c^t)_{t\geqslant 0}$ *est une suite de variables aléatoires à valeurs dans l'espace des configurations* $S^{\mathbb{T}}$*.*

Nous dirons que la dynamique (D_t) *est* uniforme *dans le temps* t *si les variables aléatoires* D_t *sont indépendantes et identiquement distribuées (i.i.d.) ; on supprime alors le* t *en indice et* $D_\delta^t(c)$ *désigne l'application de* t *mises-à-jour de la configuration* c *suivant la règle de transition* δ *et la dynamique* D*.*

Nous considérerons ici trois dynamiques particulières, toutes les trois uniformes.

Définition 2.4 (Dynamiques synchrone et asynchrones). *On appelle :*

- Dynamique synchrone, *la dynamique uniforme déterministe constante* $D = \mathbb{T}$ *mettant à jour toutes les* N *cellules à chaque pas de temps.*
- Dynamique α-asynchrone, $0 < \alpha \leqslant 1$, *la dynamique uniforme* A_α *où chaque cellule* i *appartient indépendamment à la variable d'ensemble aléatoire* A_α *avec probabilité* α*.*[28] *Sous cette dynamique,* $\alpha \cdot N$ *cellules sont mises à jour en moyenne à chaque pas de temps. Notons que la dynamique synchrone correspond à la dynamique* α*-asynchrone lorsque* $\alpha = 1$*.*
- Dynamique totalement asynchrone, *la dynamique uniforme* A_0 *consistant en un singleton aléatoire* $\{i\}$ *où* i *est choisi aléatoirement et uniformément dans l'ensemble des cellules* $\mathbb{T}$*.*[29] *Sous cette dynamique, une unique cellule aléatoire est mise à jour à chaque pas de temps. Nous verrons que* pour certaines règles de transitions*, la dynamique* A_0 *peut être vue comme une limite de la dynamique* A_α *lorsque* $\alpha \to 0$ *(mais c'est faux en général !).*

Les dynamiques synchrone et totalement asynchrone sont parfois appelées respectivement parallèle *et* séquentielle aléatoire *dans la littérature.*

Le nombre de transitions par unité de temps dépend naturellement de la fréquence des mise-à-jour, α. Afin de faciliter la lecture des figures, il est utile de renormaliser le temps.

Définition 2.5 (Temps normalisé). *On définit comme suit* $\hat{t}$*, le* temps normalisé *associé à une date* t *sous une dynamique* D*, synchrone,* α*- ou totalement asynchrone :*

[28] Cette dynamique est la dynamique habituelle des articles de la communauté "automates cellulaires probabilistes", v. p. ex. [127, 128, 129].

[29] Une autre façon de voir cette dynamique est de munir chaque cellule d'une horloge de Poisson indépendante de paramètre 1, indiquant le temps entre sa dernière mise-à-jour et la suivante. Les théorèmes classiques (v. p. ex. [113]) démontrent que l'ordre de mise-à-jour des cellules suit alors la même distribution que celle où la cellule mise à jour est tirée au hasard à chaque pas de temps (modulo une renormalisation du temps d'un facteur n). Cette interprétation permet en particulier de définir cette dynamique sur des configurations infinies. Cette dynamique est la dynamique habituelle des articles de la communauté des particules en interactions, v. p. ex. [199, 12].

- $\hat{t} = t$, *en dynamique synchrone ;*
- $\hat{t} = \alpha \cdot t$, *en dynamique α-asynchrone ; et*
- $\hat{t} = t/N$, *en dynamique totalement asynchrone.*

Cette définition assure qu'entre deux dates normalisées consécutives $\hat{t}$ et $\hat{t}+1$, exactement N cellules ont été mises à jour en espérance, quelque soit la dynamique considérée. Ceci permettra de mettre en évidence certains changements brutaux de comportement dépendant de la dynamique appliquée (p. ex., le fait que pour certaines règles, le comportement totalement asynchrone n'est pas la limite du comportement α-asynchrone pour $\alpha \to 0$, v. section 3.5).

Il est souvent commode de visualiser le comportement d'un automate cellulaire. La représentation suivante a permis de concevoir graphiquement des automates pour résoudre des tâches particulières, la synchronisation décentralisée de cellules en régime synchrone par exemple [311, 213]. Nous verrons qu'elle est également très utile pour visualiser les marches aléatoires émergeant en régime asynchrone.

Définition 2.6 (Diagramme espace temps normalisé). *Étant donné un automate cellulaire $(\mathbb{T}, S, V, \delta)$ en dynamique D, le* diagramme espace-temps normalisé *de l'automate depuis la configuration c^0 est le dessin (aléatoire) $f : \mathbb{T} \times \mathbb{R} \to S$ où le point (i, τ) a pour "couleur" $f(i, \tau) = c_i^t$ où t est l'unique instant tel que $\hat{t} \leqslant \tau < \widehat{t+1}$.*

Les figures 2.5 et 2.6 donnent des exemples de diagrammes espace-temps normalisés observés en régime asynchrone pour les automates cellulaires élémentaires (v. section 3).

2.3 Convergence

Dans la perspective d'explorer l'adéquation entre simulations et distribution limites, nous nous intéressons naturellement au temps de convergence d'un automate vers une configuration stable (la situation la plus courante en régime asynchrone).

Définition 2.7 (Configuration stable). *Nous dirons qu'une configuration c est* stable *pour un automate cellulaire $(\mathbb{T}, S, V, \delta)$, si quelque soit l'ensemble de cellules mises à jour, la configuration reste inchangée, i.e., si pour tout i, $c_i = \delta(c_{i+\partial_1}, \ldots, c_{i+\partial_\Delta})$. Notons que la notion de stabilité est indépendante de la dynamique choisie (les configurations stables sont les mêmes quelque soit la dynamique, tant que toutes les cellules ont une probabilité positive d'être mises à jour).*

Définition 2.8 (Convergence et temps de relaxation). *Nous dirons qu'un automate cellulaire $(\mathbb{T}, S, V, \delta)$* converge *en dynamique D depuis une configuration initiale $c = c^0$ si la variable aléatoire $T_c = \min\{t : c^t = D_\delta^t(c)$ est stable$\}$ est finie avec probabilité 1.*

Nous noterons $T_\delta = \max_c \mathbb{E}[T_c]$, l'espérance du temps de convergence depuis la pire des configurations initiales, appelée temps de relaxation *de l'automate $(\mathbb{T}, S, V, \delta)$. Le* temps de relaxation normalisé, *noté $\hat{T}_\delta$, se définit naturellement par $\hat{T}_\delta = \max_c \mathbb{E}[\widehat{T}_c]$.*

Comme toujours en informatique, nous nous intéressons aux comportements asymptotiques de ces quantités. Nous considérons ici essentiellement deux paramètres : le nombre de cellules N, qui pourra tendre vers l'infini ; et le taux d'asynchronisme α qui tendra soit vers 0, soit vers 1. Les relations $O(\cdot)$, $\Theta(\cdot)$, $\Omega(\cdot)$, $o(\cdot)$, $\omega(\cdot)$ qualifieront ces comportements asymptotiques pour $N \to \infty$ ou $\alpha \to 0$ ou $\alpha \to 1$: p. ex., $T = O(\frac{N}{\alpha(1-\alpha)})$ signifie qu'il

existe deux constantes $A > 0$ et $N_0 > 0$ telles que pour tout $N \geqslant N_0$ et tout $0 < \alpha < 1$, $T \leqslant A \cdot \frac{N}{\alpha(1-\alpha)}$.

Nous verrons par la suite que l'étude du temps de relaxation permet non seulement de mesurer l'adéquation entre résultats expérimentaux et théoriques, mais aussi de classifier complètement le comportement de chaque automate. En effet, il s'avère qu'au moins pour les cas que nous avons étudiés jusqu'à présent, les automates présentant des temps de relaxations identiques ont des diagrammes espace-temps normalisés mus par les mêmes processus probabilistes sous-jacents.

2.4 Cellules et transitions actives

En dynamique asynchrone, les transitions de chaque cellule peuvent être activées individuellement. Aussi il est intéressant de regarder l'action *locale* de la règle de transition de l'automate sur son voisinage. En fait, nous verrons que c'est une étape essentielle dans l'analyse de ces automates en régime asynchrone.

Définition 2.9 (Cellule active). *Nous dirons qu'une cellule i d'une configuration c est* active *pour une règle de transition δ si son état changerait si elle était mise à jour, i.e. si $c_i \neq \delta(c_{i+\partial_1}, \ldots, c_{i+\partial_\Delta})$.*

Remarquons qu'une configuration est stable si et seulement si elle n'a pas de cellule active. Lorsque la règle de transition dépend de l'état de la cellule où elle est appliquée, c.-à-d. si $\partial_1 = 0$ (ce qui est le cas des automates que nous étudions ici), nous pouvons définir la notion de transition active.

Définition 2.10 (Transition active). *La fonction de transition d'un automate est entièrement définie par l'ensemble de ses $(\#S)^\Delta$ transitions $(\eta_1, \ldots, \eta_\Delta) \mapsto \delta(\eta_1, \ldots, \eta_\Delta)$ pour tous les $(\eta_1, \ldots, \eta_\Delta) \in S^\Delta$. En supposant que η_1 réfère à l'état de la cellule mise à jour (c.-à-d., $\partial_1 = 0$), nous dirons que la transition $(\eta_1, \ldots, \eta_\Delta) \mapsto \eta'$ est* active *si $\eta_1 \neq \eta'$. Dans ce cas, nous dirons par extension que le voisinage $(\eta_1, \ldots, \eta_\Delta)$ est* actif.

Remarquons que si l'automate est à deux états $\{0, 1\}$ (ce qui sera le cas pour les automates étudiés ici), cet automate est entièrement déterminé par l'ensemble de ses transitions ou de ses voisinages actifs (les autres étant inactifs).

Pour évaluer la progression de la configuration vers un état stable, il est utile de séparer les transitions qui pourront être inversées de celles pour lesquelles une modification du voisinage est nécessaire. Les premières engendrent typiquement des comportements de marches aléatoires, tandis que les secondes marquent souvent une avancée vers une configuration stable.

Définition 2.11 (Transitions réversibles). *Considérons un automate cellulaire tel que $\partial_1 = 0$. Nous dirons qu'une transition $(\eta_1, \ldots, \eta_\Delta) \mapsto \eta'$ de cet automate est* réversible *si cet automate admet la transition inverse $(\eta', \eta_2, \ldots, \eta_\Delta) \mapsto \eta_1$. Sinon, nous dirons que la transition est* irréversible.

Le logiciel que nous avons développé pour explorer les comportements asynchrones des automates 2D [283] permet de visualiser les cellules actives sous la forme d'une pastille de couleur rouge pour les réversibles et violette pour les irréversibles. Toutes les illustrations de ce manuscrit ont été obtenues via ce logiciel.

2.5 Automates cellulaires élémentaires et totalisants

Les automates que nous considérons dans ce manuscrit auront pour ensemble d'états $S = \{0, 1\}$ où 0 et 1 seront représentés par les couleurs blanche et noire respectivement. Nous étudions principalement trois topologies d'automates :

- *élémentaire :* $((\mathbb{T}, +), V) = ((\mathbb{Z}_n, +), \{-1, 0, 1\})$, le tore unidimensionnel avec la topologie des plus proches voisins.
- *von Neumann 2D :* $((\mathbb{T}, +), V) = ((\mathbb{Z}_n \times \mathbb{Z}_m, +), \{(0,0), (1,0), (0,1), (-1,0), (0,-1)\})$, le tore bidimensionnel avec la topologie des quatre plus proches voisins.
- *Moore 2D :* $((\mathbb{T}, +), V) = ((\mathbb{Z}_n \times \mathbb{Z}_m, +), \{(0,0), (1,0), (1,1), (0,1), (-1,1), (-1,0), (-1,-1), (0,-1), (1,-1)\})$, le tore bidimensionnel avec la topologie des huit plus proches voisins.

Lorsque la topologie est élémentaire, pour toute configuration $c \in \{0, 1\}^n$ et tout motif $w \in \{0, 1\}^+$ de longueur $|w| = \ell$, on note $|c|_w = \#\{i : c_i \dots c_{i+\ell-1} = w_1 \dots w_\ell\}$ le nombre d'occurrences du motif w dans la configuration c. Par exemple, $|101001010|_{01} = 4$.

2.5.1 Automates cellulaires élémentaires (1D)

Les automates cellulaires à deux états ayant la topologie élémentaire sont appelés *automates cellulaires élémentaires*. Il est commode de pouvoir référencer les automates par des numéros.

Notation 2.12 (Notation traditionnelle des automates cellulaires élémentaires). *Il y a $8 = 2^3$ voisinages possibles et donc 2^8 automates cellulaires élémentaires au total caractérisés par leur fonction de transition δ. Wolfram a proposé la numération fort commode suivante [323]. Chaque voisinage est codé par un entier sur 3 bits correspondant aux valeurs des trois états, le voisinage de code 3 désigne le triplet d'états $(0, 1, 1)$. La fonction de transition δ est alors codée par un entier r_δ sur 8 bits, compris entre 0 et 255, où le $v^{\text{ème}}$bit est l'image du voisinage de code v par δ :*

$$r_\delta = \sum_{v=0}^{7} \delta(v_{-1}, v_0, v_1) \cdot 2^v, \quad \text{où } v_j \text{ désigne le } (1-j)^{\text{ème}} \text{bit de } v.$$

Par exemple, l'automate élémentaire `nul` où chaque cellule mise-à-jour passe dans l'état 0, est codé par l'entier **0**, et l'automate élémentaire `identité` est codé par $2^{010} + 2^{011} + 2^{110} + 2^{111} = 2^2 + 2^3 + 2^6 + 2^7 =$ **204**. L'avantage de cette notation est qu'elle est très facile à coder et décoder.

Comme nous l'avons évoqué précédemment, en régime asynchrone, il est utile de s'intéresser à l'action locale de l'automate sur la configuration. Aussi, il sera plus commode d'identifier les automates cellulaires par l'ensemble de leurs transitions actives. Comme les automates cellulaires sont à deux états $\{0, 1\}$, il y a exactement huit transitions actives possibles (autant que de voisinages) : $(v_{-1}, v_0, v_1) \mapsto 1 - v_1$ pour $v = 0$ à 7 où v_j désigne le $(1-j)^{\text{ème}}$ bit de v. Afin d'éviter la confusion entre les deux notations, nous étiquetons les différentes transitions actives par les lettres **A** à **H** correspondant aux voisinages de 0 à 7 (illustration figure 2.3) : la lettre correspondant au voisinage v est obtenue simplement par la formule `ascii`$(65 + v)$. Un automate élémentaire de règle δ sera alors identifié par l'ensemble Λ_δ des lettres correspondant à ses transitions actives.

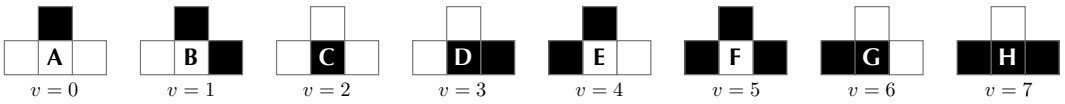


Figure 2.3 – Notation des différentes transitions actives. *(on note traditionnellement le nouvel état de la cellule centrale au-dessus de son voisinage, à la manière d'un mini-diagramme espace temps où le temps s'écoule vers le haut)*

Remarquons qu'on passe de la notation de Wolfram r_δ à la notation par lettres Λ_δ de la façon suivante :

$$\begin{cases} r_\delta & \mapsto & \Lambda_\delta = \{\texttt{ascii}(65+v) : 0 \leqslant v \leqslant 7 \text{ tel que } (2^{\text{ème}} \text{ bit de } v) \neq (v^{\text{ème}} \text{ bit de } r_\delta)\} \\ \Lambda_\delta & \mapsto & r_\delta = \sum_{a\in\{\texttt{A,B,E,F}\}} \mathbb{1}_{a\in\Lambda_\delta}\, 2^{\text{int}(a)-65} + \sum_{a\in\{\texttt{C,D,G,H}\}} \mathbb{1}_{a\notin\Lambda_\delta}\, 2^{\text{int}(a)-65} \end{cases}$$

Par exemple, l'automate `identité` a pour code de Wolfram **204** et est codé par l'ensemble vide $\varnothing$ puisqu'aucune transition n'est active. L'automate `nul` a pour code de Wolfram **0** et est codé par les lettres **CDGH** puisque ses seuls voisinages sont ceux où la cellule centrale est à `1`.

Remarque 2.13. La convention que nous adoptons dans ce document est *différente* de la notation par lettres initialement proposée par Nazim Fatès dans sa thèse [107] et dans nos articles [110, 112, 111]. Nous avons abandonnée cette dernière car elle n'était pas facile à décoder et donc très difficile à mémoriser.

Comme il est commode de disposer des deux notations pour se référer à un automate, nous les utiliserons ensemble. Par exemple, les automates `identité` et `nul` seront notés $\varnothing$(**204**) et **CDGH**(**0**) respectivement.

2.5.2 Automates cellulaires totalisants externes 2D

Dans le cas des automates cellulaires 2D à deux états, il y a :

- pour la topologie de von Neumann : $2^5 = 32$ voisinages possibles, soit $2^{32} = 4\,294\,967\,296$ automates cellulaires différents (plus de quatre milliards).
- pour la topologie de Moore : $2^9 = 512$ voisinages possibles, soit $2^{512} \approx 10^{156}$ automates cellulaires différents, soit près du carré du nombre supposé d'atomes dans l'univers... (avec deux états seulement !)

Il semble donc désespéré d'explorer l'ensemble de ces automates, même avec deux états seulement. Aussi, nous nous intéresserons à une classe d'automates 2D particulière, que nous espérons accessible : les totalisants externes, pour lesquels la transition se détermine à partir de son état et du nombre de ses voisins dans chacun des états. Ces automates ont l'avantage d'avoir un sens physique/biologique si l'on pense en terme de concentrations. Cette classe d'automates joue d'ailleurs un rôle important dans la définition des réseaux de neurones qui utilisent des automates à seuil, un cas particulier d'automates totalisants externes.

Définition 2.14 (Automates totalisants externes). *Un automate cellulaire $(\mathbb{T}, S, V, \delta)$ à deux états $S = \{0, 1\}$ est* totalisant externe *s'il existe une fonction $\sigma : S \times \{0, \ldots, \Delta - 1\} \to S$ telle que pour tout voisinage $(v_0, v_1, \ldots, v_{\Delta-1})$,*

$$\delta(v_0, v_1, \ldots, v_{\Delta-1}) = \sigma(v_0, v_1 + \cdots + v_{\Delta-1}),$$

où v_0 désigne l'état de la cellule mise à jour.

Il y a donc :

- pour la topologie de von Neumann : $2^{2\times 5} = 1\,024$ automates totalisant externes.
- pour la topologie de Moore : $2^{2\times 9} = 262\,144$ automates totalisant externes.

Ces automates restent nombreux, mais leur nombre diminue significativement en tenant compte des symétries. On peut donc espérer en analyser les comportements. De la même manière que précédemment, on peut numéroter commodément les automates totalisant de la façon suivante. Comme précédemment, on numérote les entrées possibles (s, Σ), avec $s \in \{0, 1\}$ et $\Sigma \in \{0, \ldots, \Delta - 1\}$, de la fonction de transition σ de 0 à $2\Delta - 1$ en les énumérant dans l'ordre :

$$(0,0), (1,0), (0,1), (1,1), (0,2), (1,2), \ldots, (0,\Delta-1), (1,\Delta-1).$$

Le code de l'automate totalisant externe est alors :

$$r_\sigma = \sum_{\Sigma=0}^{\Delta-1} (2\,\sigma(1,\Sigma) + \sigma(0,\Sigma)) \cdot 2^{2\Sigma}$$

De nouveau, cette notation présente l'avantage d'être très facile à coder et décoder. Par exemple, si l'on considère les automates Majorité et Minorité pour les topologies de von Neumann et Moore, on obtient les codes suivants :

σ : Majorité *(von Neumann)*

s \ Σ	4	3	2	1	0
0	1	1	0	0	0
1	1	1	1	0	0

Code = $2^5 + 2^6 + \cdots + 2^9 =$ **992**

σ : Minorité *(Moore)*

s \ Σ	8	7	6	5	4	3	2	1	0
0	0	0	0	0	1	1	1	1	1
1	0	0	0	0	0	1	1	1	1

Code = $2^0 + 2^1 + \cdots + 2^8 =$ **511**

Nous utiliserons dans la suite de ce manuscrit les notations **OTN992** et **OTM511** pour désigner les automates totalisants externes n°992 pour la topologie de von Neumann et n°511 pour la topologie de Moore, respectivement. Cette convention est celle de notre simulateur d'automates totalisants externes 2D [283].

3 Études en dimension 1

3.1 Automates cellulaires élémentaires doublement quiescents (DQECA)

Nous nous sommes tout d'abord intéressés à une sous-classe particulière des automates élémentaires, les automates doublement-quiescents.

Définition 2.15 (doublement-quiescent). *Un automate cellulaire élémentaire est* doublement-quiescent *(un* DQECA*, en abrégé) si les voisinages tout-noir et tout-blanc sont inactifs, i.e. si son code exclut les lettres* **A** *et* **H**.

Une motivation pour l'étude de cette sous-classe d'automates est qu'une cellule ne va pas changer d'état spontanément mais uniquement lorsqu'elle y est contrainte par son environnement, ce qui semble une hypothèse raisonnable pour modéliser, p. ex., des phénomènes non-bruités. C'est aussi une classe d'automates qui présente des propriétés intéressantes qui facilitent leur analyse.

Tous les DQECA admettent pour configurations stables les configurations toute blanche et toute noire, que nous appellerons configurations *triviales*. Certaines règles doublement quiescentes admettent des configurations stables non-triviales :

Propriété 2.16 (Configurations stables des DQECA). *Soit c une configuration stable d'un DQECA.*

- *si les transitions* **B** *ou* **E** *sont actives, alors tous les* 0 *de c sont isolés.*
- *si les transitions* **D** *ou* **G** *sont actives, alors tous les* 1 *de c sont isolés.*
- *si la transition* **F** *est active, alors aucun* 0 *de c n'est isolé.*
- *si la transition* **C** *est active, alors aucun* 1 *de c n'est isolé.*

Propriété 2.17 (Symétries). *Il existe exactement 24 DQECA distincts une fois appliquées les symétries* 0/1 *et horizontale dans la règle de transition. Ils sont listés dans les figures et avec les différents DQECA de leur classe en gris entre crochets.*

Notons que lorsque la configuration est infinie ou de longueur paire, on peut considérer une autre symétrie, celle qui consiste à prendre le XOR (noté $\oplus$) de la configuration avec le motif $(01)^\omega$. Cette opération produit un nouvel automate cellulaire s'il existe un ECA δ' tel que pour toute configuration infinie c, $\delta(c) \oplus (01)^\omega = \delta'(c \oplus (01)^\omega)$. Or cette condition est équivalente à : pour tout voisinage v codé sur 3 bits, $\delta(v) = 1 \oplus \delta(v \oplus 111)$, ou encore au fait que les voisinages **A**, **B**, **C** et **D** sont actifs si et seulement si leurs conjugués respectifs **H**, **G**, **F** et **E** le sont. Il y a donc exactement seize automates symétrisables par XOR du motif $(01)^\omega$ (dont huit DQECA) et leur ECA symétrique a pour règle : $\delta'(v) = \delta(v \oplus 101)$ pour tout voisinage v.

L'application de cette symétrie peut activer les transitions **A** et **H**. On en déduit que nos résultats sur les DQECA se transportent à quatre autres automates : les automates **ABGH**(**15**), **ABDEGH**(**23**), **AH**(**77**) et **ADEH**(**85**), les symétriques par XOR $(01)^\omega$ des DQECAs **BCFG**(**170**), **BCDEFG**(**178**), **CF**(**232**) et **CDEF**(**240**) respectivement (les automates **BDEG**(**150**), ∅(**204**)sont leurs propres symétriques et **BG**(**142**) et **DE**(**212**) sont symétriques l'un de l'autre). Ces symétriques sont indiqués entre chevrons dans les cases correspondantes à leur représentant dans les figures 2.5 et 2.6. Parmi ces automates, la règle de minorité **ABDEGH**(**23**)nous intéressera particulièrement à la section 4.

3.2 Simulations

Les figures 2.5 et 2.6 présentent le résultat des simulations des vingt-quatre représentants des DQECA. La première colonne contient les numéro et nom de l'automate et des DQECA qui lui sont équivalents. Les onze suivantes présentent ses diagrammes espace-temps normalisés obtenus par simulations sur des configurations toriques de longueur $n = 50$ pendant une durée de 100 en temps normalisé à partir de configurations initiales aléatoires (chaque cellule est à 1 avec probabilité $\frac{1}{2}$ indépendamment), en dynamique α-asynchrone pour α variant de 0 (régime totalement asynchrone) à 1 (régime synchrone) par pas de 0.1. La dernière colonne représente l'évolution de la densité de la configuration. Comme les DQECA ont tendance à

converger vers les configurations toute-blanche et toute-noire, et que la densité moyenne ne permet pas de distinguer un automate divergeant dont la densité reste autour de 50% d'un automate convergeant avec probabilité égale vers tout-blanc ou tout-noir, nous avons décidé de mesurer, plutôt que la densité, la distance de la configuration aux deux configurations extrêmes 0^n et 1^n. Nous définissons donc l'*écart de densité* $\xi(c)$ d'une configuration c comme la distance de sa densité à 0 ou 1, c.-à-d. :

$$\xi(c) = \frac{1}{2} - \left| \frac{|c|_1}{n} - \frac{1}{2} \right| \quad \text{(v. figure 2.4)}$$

Ainsi, $\xi(c) = 0 \Leftrightarrow c \in \{0^n, 1^n\}$, et $\xi(c) = \frac{1}{2}$ signifie que c a pour densité $\frac{1}{2}$. $\xi(c)$ est donc bien une mesure de distance de c aux deux configurations stables 0^n et 1^n. En particulier, dans le cas d'un automate qui converge avec probabilité égale vers 0^n ou 1^n, l'espérance de son écart de densité vaudra 0, rendant compte ainsi correctement de la convergence de l'automate.

La treizième colonne des figures 2.5 et 2.6 donnent les valeurs moyennes sur cent runs de l'écart de densité pour chaque automate et chaque valeur de α depuis des configurations initiales aléatoires, à différentes dates normalisées $\hat{t} \in \{5, 10, 50, 100, 500, 1000, 1500, 2000, 2500\}$ traçant l'évolution de la configuration vers les configurations 0^n et 1^n au fur et à mesure que le temps avance. Les courbes correspondant à chaque date normalisée sont tracées de couleurs différentes allant du rouge pour $\hat{t} = 5$ au violet pour $\hat{t} = 2500$. Ces couleurs permettent de visualiser la rapidité de la convergence, en comptant le nombre de courbes qui ne sont pas confondues avec la courbe violette finale.

La première constatation issue de ces simulations est que les diagrammes espace-temps des DQECA en régime asynchrone diffèrent radicalement de leurs diagrammes espace-temps synchrones. En particulier, la quasi-totalité de ces automates convergent en régime asynchrone alors qu'ils divergent majoritairement en régime synchrone. Ainsi, conformément à ce qui avait été observé dans des cas particuliers par [58, 45, 286, 109], pour cette topologie, les mises-à-jour asynchrones semblent faciliter la convergence des automates cellulaires.

L'automate identité ∅(**204**) (le dernier de la figure 2.6) est donné comme référence pour la valeur de l'écart de densité des configurations aléatoires. Les couleurs des courbes de densités permettent de visualiser essentiellement quatre vitesses de convergence :

- *quasi-instantannée* (▶▶▶) : toutes les courbes sont confondues avec la courbe violette qui se situe à un niveau bas ;
- *rapide* (▶▶) : on distingue une ou deux courbes (couleurs rouge et orange) de la courbe violette, située à un niveau très bas ;

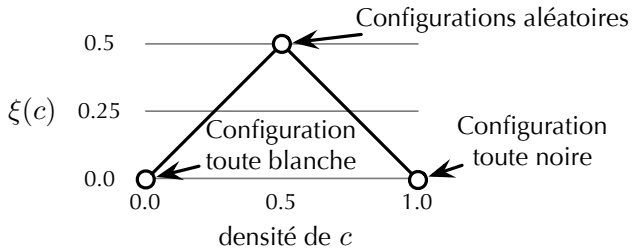


Figure 2.4 – l'écart de densité ξ mesure une distance aux deux configurations 0^n et 1^n.

- *moyenne* (▶) : on distingue toutes les couleurs des courbes et la courbe violette est située très bas ;
- *non-observable* (▷) ou *divergent* (∞) : toutes les courbes sont confondues avec la courbe violette située à un niveau moyen ou haut.

Ces couleurs permettent donc de distinguer très nettement les différents comportements asynchrones des DQECA, présentés dans le tableau 2.1.

Automates cellulaires	Convergence observée $\alpha = 0$	$0 < \alpha < 1$	$\alpha = 1$
C(**200**), **CF**(**232**)	▶▶▶	▶▶▶	▶▶▶
CDG(**128**), **DG**(**132**), **CG**(**136**), **G**(**140**), **CDFG**(**160**), **DFG**(**164**), **FG**(**172**)	▶▶	▶▶	▶▶
BCDG(**130**), **BCDFG**(**162**)	▶▶	▶▶	∞
BCD(**194**)	▶	▶▶	∞
BCG(**138**), **BCFG**(**170**), **CEFG**(**184**)	▶	▶	∞
BCDEG(**146**), **BCDEFG**(**178**)	▶	▶ pour $\alpha < \alpha_c$ ▷ pour $\alpha > \alpha_c$ $\alpha_c^{\mathbf{146}} \approx 0.66$ $\alpha_c^{\mathbf{178}} \approx 0.50$	∞
DEFG(**180**)	▷	▶	∞
BDG(**134**)	∞	▶	∞
BDEG(**150**), **BG**(**142**), **EG**(**156**)	∞	∞	∞

Table 2.1 – Vitesses de convergence observées pour les différents DQECA asynchrones.

On constate que malgré la simplicité de leur description, les DQECA présentent déjà une grande richesse de comportements. La suite de cette section est consacrée à la preuve de ces observations. Nous verrons que dans certains cas (principalement les automates **BCDEG**(**146**), **BCDEFG**(**178**), **DEFG**(**180**) et **BDG**(**134**)), des phénomènes fort intriqués apparaissent et font que nous sommes actuellement incapables d'analyser complètement leurs comportements.

3.3 Boîte à outils probabilistes

Nos preuves sont basées sur la construction de couplages probabilistes. Étant données deux variables aléatoires X et Y, un *couplage probabiliste* des variables X et Y est une variable aléatoire $Z = (X, Y)$ dont les lois marginales sont celles de X et de Y. Les variables sont dites couplées car leurs valeurs ne sont pas indépendantes.

Dans notre cas, nous cherchons à coupler un processus probabiliste (X_t) avec une suite de variables aléatoires (Φ_t) à valeurs réelles, que nous appelerons *variant*,[30] de façon que le processus (X_t) ait convergé avant la date t si $\Phi_t = 0$. On pourra alors majorer l'espérance du temps de convergence de (X_t) par l'espérance du premier temps d'atteinte de 0 par (Φ_t).

[30] On parle aussi de *fonction de Lyapunov*.

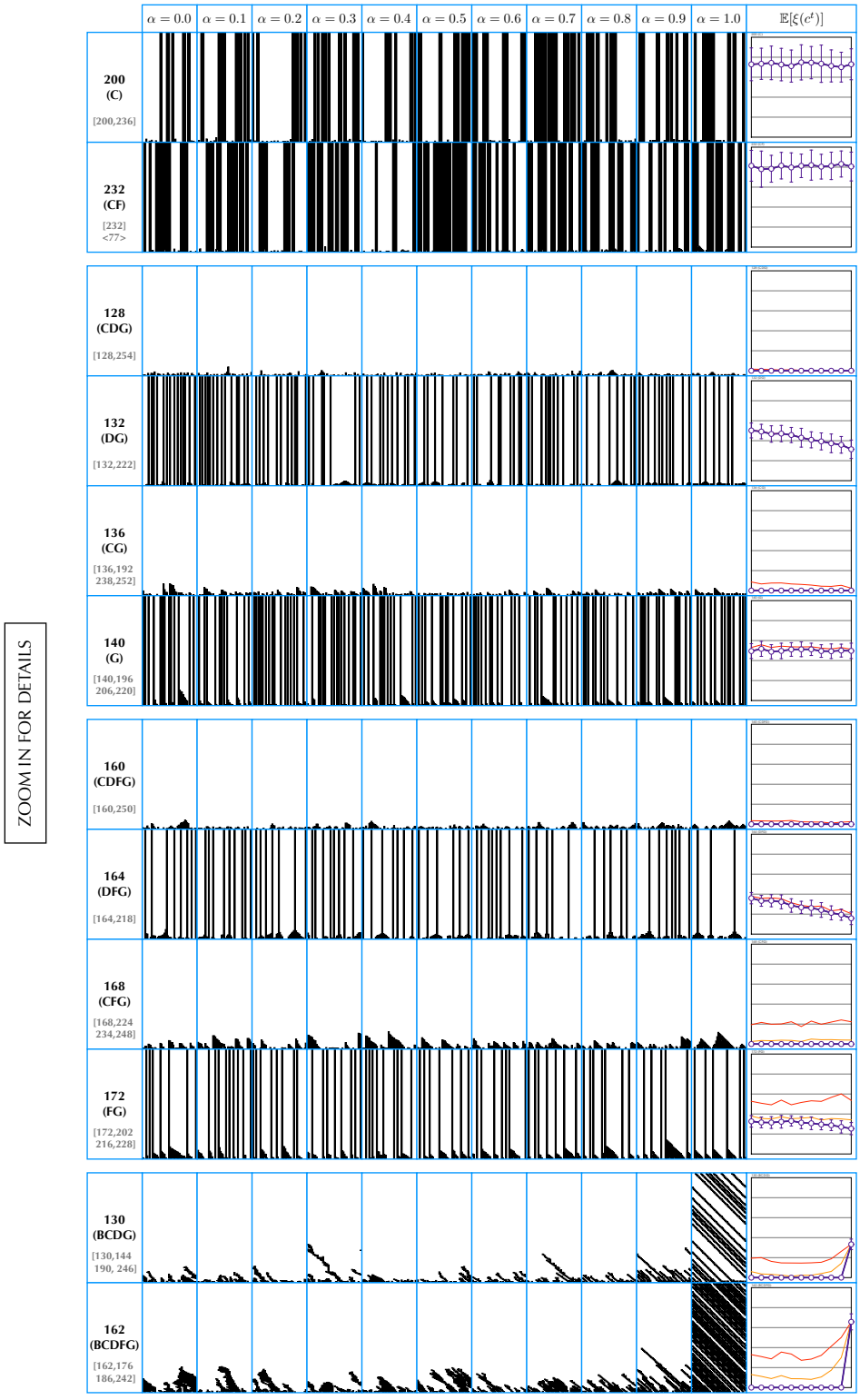


Figure 2.5 – Diagrammes espace-temps des représentants des DQECA. Le graphe à droite représente les moyenne et écart-type de l'écart de densité $\xi(c^t)$ pour $\alpha \in \{0, 0.1, \ldots, 0.9, 1.0\}$ sur 100 *runs* depuis des configurations initiales aléatoires uniformes avec $n = 50$. Les différentes courbes correspondent aux différentes dates normalisées $\hat{t}$ des mesures : $\hat{t} = 5$ en rouge, $\hat{t} = 10$ en orange, $\hat{t} = 50$ en jaune, $\hat{t} = 100$ en vert anis, $\hat{t} = 500$ en vert d'eau, $\hat{t} = 1000$ en bleu ciel, $\hat{t} = 1500$ en bleu roi, $\hat{t} = 2000$ en violet clair, et $\hat{t} = 2500$ en violet foncé.

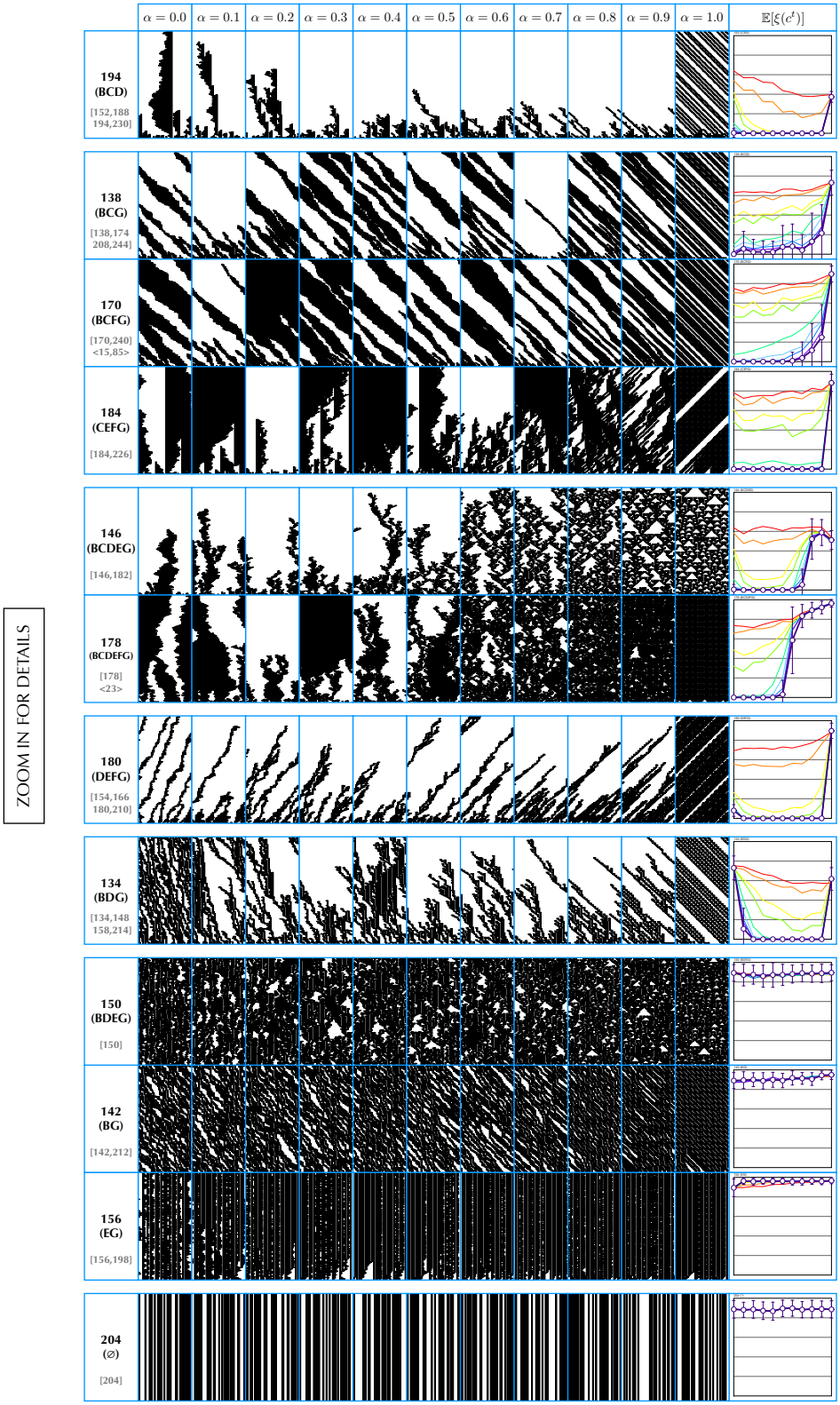


Figure 2.6 – Diagrammes espace-temps des représentants des DQECA. Le graphe à droite représente les moyenne et écart-type de l'écart de densité $\xi(c^t)$ pour $\alpha \in \{0, 0.1, \ldots, 0.9, 1.0\}$ sur 100 *runs* depuis des configurations initiales aléatoires uniformes avec $n = 50$. Les différentes courbes correspondent aux différentes dates normalisées $\hat{t}$ des mesures : $\hat{t} = 5$ en rouge, $\hat{t} = 10$ en orange, $\hat{t} = 50$ en jaune, $\hat{t} = 100$ en vert anis, $\hat{t} = 500$ en vert d'eau, $\hat{t} = 1000$ en bleu ciel, $\hat{t} = 1500$ en bleu roi, $\hat{t} = 2000$ en violet clair, et $\hat{t} = 2500$ en violet foncé.

En pratique, on construit un couplage entre deux variables aléatoires en les faisant travailler sur le même espace probabilisé. Considérons par exemple une suite de configurations unidimensionnelles (c^t) dont on souhaite étudier le temps de convergence. En admettant que la configuration nulle 0^n soit stable, si l'on est capable de démontrer que la variable $\Phi_t = |c^t|_1$ comptant le nombre de 1 dans la configuration c^t, décroît d'au moins $\varepsilon > 0$ en espérance à chaque pas de temps tant que c^t n'a pas atteint une configuration stable, on obtient que l'espérance du temps de convergence de c^t est inférieur à Φ_0/ε puisque si $\Phi_t = 0$, $c^t = 0^n$ est une configuration stable. Dans ce couplage $Z_t = (c^t, \Phi_t)$, la première loi marginale est celle du processus que l'on souhaite étudier, et la seconde est majorée stochastiquement par celle d'une marche aléatoire positive biaisée vers 0. La seconde variable, plus simple, que l'on peut étudier indépendamment de la première, nous permet de conclure sur les propriétés de la première. Remarquons qu'il est tout à fait possible que c^t ait convergé avant que Φ_t n'atteigne 0. Nous verrons des couplages plus sophistiqués aux sections 3.4 et 4.3 (v. également [255, 258]).

Il est souvent utile de conditionner par rapport au passé. Lorsque l'on utilise un couplage, le passé doit englober également le passé de l'autre variable pour être cohérent. Cette notion est formalisée en probabilités par celle de *filtration adaptée* $(\mathcal{F}_t)$, qui consiste en la sous-σ-algèbre engendrée par les événements du passé par rapport auxquels on souhaite conditionner nos variables (v. [64, 153]). Typiquement, dans ce document, la filtration adaptée sera la sous-σ-algèbre engendrée par le conditionnement aux $t+1$ premières valeurs de la configuration $c^0, \dots, c^t$.

Notation 2.18. *Étant donnée une suite de variables aléatoires $(\Phi_t)_{t\geqslant 0}$ à valeurs réelles, on note $\Delta\Phi_t = \Phi_t - \Phi_{t-1}$, pour $t \geqslant 1$, la variable aléatoire ayant pour valeur la variation de Φ_t entre $t-1$ et t.*

Les lemmes suivants formalisent le raisonnement exposé dans l'exemple ci-dessus.

Lemme 2.19 (Cas monotone, [111]). *Soient $\varepsilon > 0$, $m, m' > 0$, et une suite de variables aléatoires (Φ_t) à valeurs dans $[-m', m]$ munie d'une filtration adaptée $(\mathcal{F}_t)$, et telle que pour tout $t \geqslant 0$,*

$$\Phi_t > 0 \Rightarrow \mathbb{E}[\Delta\Phi_{t+1}|\mathcal{F}_t] \leqslant -\varepsilon.$$

Soit $T = \min\{t : \Phi_t \leqslant 0\}$ la variable aléatoire du premier temps tel que $\Phi_t \leqslant 0$. Alors,

$$\mathbb{E}[T] \leqslant \frac{m' + \mathbb{E}[\Phi_0]}{\varepsilon}.$$

La preuve de ce lemme consiste à démontrer que $\mathbb{E}[T] < \infty$ à l'aide de l'inégalité de Markov puis à appliquer un théorème du temps d'arrêt optionnel (v. [153]) au temps d'arrêt T pour la surmartingale $Y_t = \Phi_t + \varepsilon \cdot t$ (v. [111]).

Étudions à présent le cas d'un variant borné dont l'espérance de la variation est nulle mais qui a une probabilité $> \varepsilon$ d'augmenter et de diminuer. Soient $\varepsilon > 0$, un entier $m > 0$ et une suite de variables aléatoires à valeurs dans $[0, m]$ munie d'une filtration adaptée $(\mathcal{F}_t)$.

Définition 2.20. *Nous dirons que* (Φ_t) *est :*[31]

- *de* type I *si pour tout* t *:*
  - $\mathbb{E}[\Phi_{t+1}|\mathcal{F}_t] = \Phi_t$ *(i.e.,* Φ_t *est une martingale) et*
  - *si* $0 < \Phi_t < m$*, alors* $\Pr\{\Delta\Phi_{t+1} \geqslant 1|\mathcal{F}_t\} \geqslant \varepsilon$ *et* $\Pr\{\Delta\Phi_{t+1} \leqslant -1|\mathcal{F}_t\} \geqslant \varepsilon$.
- *de* type II *si pour tout* t *:*
  - *si* $\Phi_t < m$*,* $\mathbb{E}[\Phi_{t+1}|\mathcal{F}_t] = \Phi_t$ *(i.e.,* Φ_t *se comporte comme une martingale quand* $\Phi_t < m$*),*
  - *si* $0 < \Phi_t < m$*, alors* $\Pr\{\Delta\Phi_{t+1} \geqslant 1|\mathcal{F}_t\} \geqslant \varepsilon$ *et* $\Pr\{\Delta\Phi_{t+1} \leqslant -1|\mathcal{F}_t\} \geqslant \varepsilon$*, et*
  - *si* $\Phi_t = m$*, alors* $\Pr\{\Phi_{t+1} \leqslant m-1|\mathcal{F}_t\} \geqslant \varepsilon$ *(i.e.,* Φ_t *rebondit sur* m*).*

Lemme 2.21 (Cas d'une martingale à variance positive, [111]). *Si* (Φ_t) *est de type I, posons* $T = \min\{t : \Phi_t \in \{0, m\}\}$*, alors :*

$$\mathbb{E}[T] \leqslant \frac{m\mathbb{E}[\Phi_0] - \mathbb{E}[\Phi_0^2]}{2\varepsilon}.$$

La preuve repose sur l'application du théorème du temps d'arrêt optionnel à la sousmartingale $Y_t = \Phi_t^2 - 2\varepsilon t$, de filtration adaptée $(\mathcal{F}_t)$, pour le temps d'arrêt d'espérance finie T (v. [111]).

Lemme 2.22 (Cas d'une "semi"-martingale à variance positive,[111]). *Si* (Φ_t) *est de type II, posons* $T = \min\{t : \Phi_t = 0\}$*, alors :*

$$\mathbb{E}[T] \leqslant \frac{(2m+1)\mathbb{E}[\Phi_0] - \mathbb{E}[\Phi_0^2]}{2\varepsilon}.$$

La preuve repose sur l'application du même théorème que précédemment mais à la sousmartingale $Y_t = \Phi_t^2 - (2m+1)\Phi_t - 2\varepsilon t$, de filtration adaptée $(\mathcal{F}_t)$ (v. [111]).

3.4 Étude du régime totalement asynchrone

Une particularité du régime totalement asynchrone des DQECA est que l'on peut décomposer toute configuration en régions de `0` et de `1` consécutifs et que le nombre de ces régions ne peut que diminuer.

Définition 2.23 (Région). *On appelle* `0`-région *(resp.,* `1`-région*) d'une configuration* c *toute suite maximale de* `0` *(resp.,* `1`*) consécutifs. On note* $Z(c) = |c|_{01}(= |c|_{10})$ *le nombre de* `0`*- ou de* `1`*-régions de la configuration.*

Propriété 2.24. *Pour tout DQECA en régime totalement asynchrone,* $Z(c^t)$ *est une variable aléatoire décroissante en fonction de* t*. De plus* $Z(c^{t+1}) < Z(c^t)$ *si et seulement si la cellule mise-à-jour au temps* t *a pour voisinage actif* **C** *ou* **F**.

Cette propriété est une conséquence immédiate du fait que les voisinages **A** et **H** sont inactifs pour un DQECA.

L'analyse des régimes transitoires repose donc sur l'étude de l'évolution des régions de la configuration. La figure 2.7 représente l'action de chaque transition sur ces régions :

[31] Ces définitions sont un peu plus générales que celles de [111] où nous imposions inutilement que $\Pr\{\Delta\Phi_{t+1} \geqslant 1|\mathcal{F}_t\} = \Pr\{\Delta\Phi_{t+1} \geqslant 1|\mathcal{F}_t\}$ lorsque Φ_t n'est pas aux bords. Les preuves des lemmes qui suivent sont inchangées et cela facilitera leurs applications par la suite.

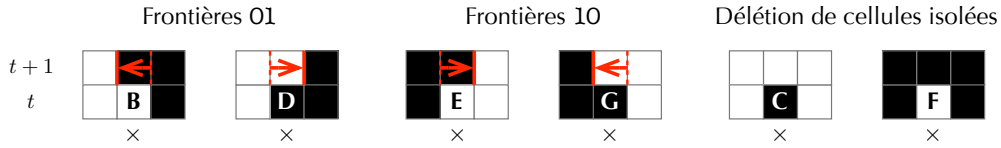


Figure 2.7 – Évolutions locales des régions pour les DQECA en régime totalement asynchrone. (la cellule mise à jour au temps t est marquée d'une croix $\times$)

- les transitions **B** et **D** déplacent les frontières 01 vers la gauche et vers la droite, respectivement ;
- les transitions **G** et **E** déplacent les frontières 10 vers la gauche et vers la droite, respectivement ; et
- les transitions **F** et **C** éliminent les 0 et les 1 isolés, respectivement, et permettent donc la fusion de deux 1- et 0-régions voisines, respectivement.

Cette lecture permet de comprendre le comportement totalement asynchrone des DQECA en fonction de leurs actions sur ces frontières. Par exemple, si les transitions **B** et **D** sont actives, les frontières 01 exécuteront des marches aléatoires. Si la transition **E** est active et pas la **G**, alors les frontières 10 se déplaceront vers la droite et les 1-régions s'étendront vers la droite à un rythme 1 en temps normalisé. Si les transitions **C** et **F** sont inactives, aucune région ne peut fusionner. Les actions de chaque représentant des DQECA sur les régions d'une configuration sont résumées sur la table 2.2. Ces comportements vont nous permettre de définir les variants qui nous serviront à borner les temps de relaxation de ces différents automates pour démontrer le théorème suivant :

Théorème 2.25 ([111]). *En dynamique totalement asynchrone, parmi les soixante-quatre DQECA,*

- *cinquante cinq convergent presque sûrement vers une configuration stable aléatoire depuis n'importe quelle configuration initiale et ont pour temps de relaxation normalisé :* 0, $\Theta(\log n)$, $\Theta(n)$, $\Theta(n^2)$ *ou* $\Theta(2^n)$ *;*
- *les neuf autres divergent presque sûrement depuis toute configuration non-triviale et différente de* $(01)^{n/2}$ *si* n *est pair.*

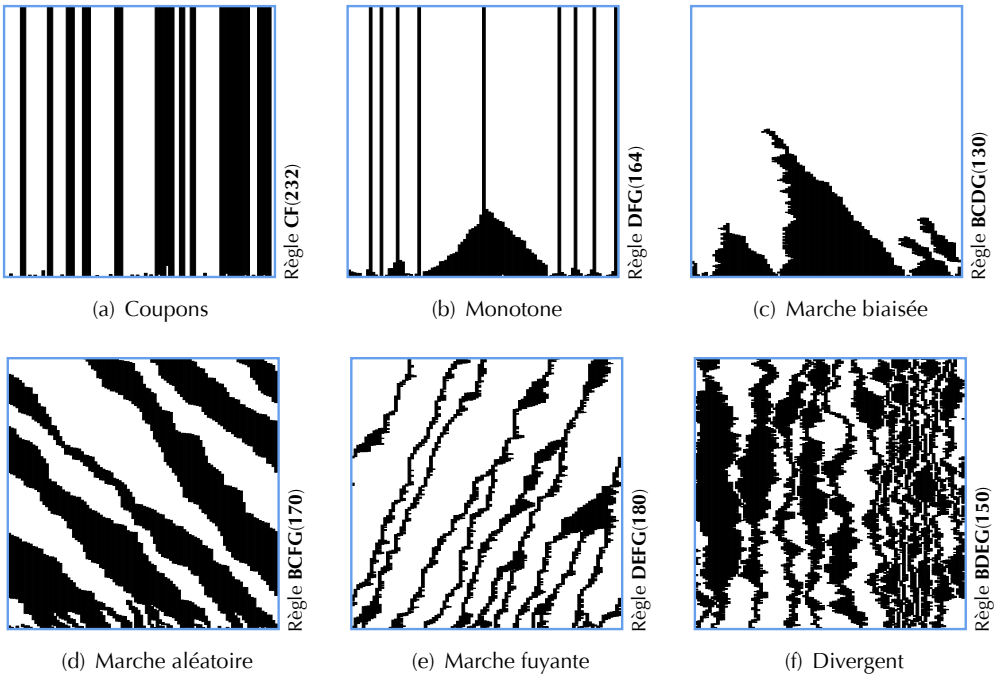
De plus, les comportements des DQECA ayant le même temps de relaxation normalisé sont identiques : inerte pour 0*, collectionneur de coupons pour* $\Theta(\log n)$ *(fig. 2.8(a)), monotone ou marche aléatoire biaisée favorablement pour* $\Theta(n)$ *(fig. 2.8(b) et 2.8(c))), marche aléatoire non-biaisée pour* $\Theta(n^2)$ *(fig. 2.8(d)) et marche aléatoire fuyante pour* $\Theta(2^n)$ *(fig. 2.8(e)). Enfin, ce comportement peut être déduit de la lecture de leur code (v. table 2.2).*

Ce résultat est obtenu en associant à chaque automate une fonction potentiel mesurant la distance de la configuration courante à une configuration stable. On démontre alors que si la fonction potentiel atteint un certain seuil (typiquement, 0 ou n), la configuration est stable et le processus a convergé. L'étude de l'évolution de la fonction potentiel sous l'action de la règle de l'automate permet de déterminer les bornes sur le temps de relaxation. Il nous a semblé inutile dans ce document de reprendre en détail les calculs pour chaque automate. Aussi, nous ne le ferons que pour deux d'entre eux : le shift **BCFG**(**170**) et l'automate **DEFG**(**180**).

Comportement	DQECA (#)	Règle	←B/D→ **01**	←G/E→ **10**	0:F,1:C Délétion	Tps de relaxation normalisé ($\widehat{\mathrm{TR}}$)
Inerte	**204**	(1) ∅	·	·	·	0
Collectionneur de coupons (fig. 2.8(a))	**200**	(2) **C**	·	·	1	$\Theta(\log n)$
	232	(1) **CF**	·	·	0,1	
Monotone (fig. 2.8(b))	**128**	(2) **CDG**	→	←	1	$\Theta(n)$
	132	(2) **DG**	→	←	·	
	136	(4) **CG**	·	←	1	
	140	(4) **G**	·	←	·	
	160	(2) **CDFG**	→	←	0,1	
	164	(2) **DFG**	→	←	0	
	168	(4) **CFG**	·	←	0,1	
	172	(4) **FG**	·	←	0	
Marche aléatoire biaisée (fig. 2.8(c))	**130**	(4) **BCDG**	↼↽	←	1	
	162	(4) **BCDFG**	↼↽	←	0,1	
Marche aléatoire (fig. 2.8(d))	**138**	(4) **BCG**	←	←	1	$\Theta(n^2)$
	146	(2) **BCDEG**	↼↽	↼↽	1	
	170	(2) **BCFG**	←	←	0,1	
	178	(1) **BCDEFG**	↼↽	↼↽	0,1	
	184	(2) **CEFG**	·	↼↽	0,1	
	194	(4) **BCD**	↼↽	·	1	
Marche aléatoire fuyante (fig. 2.8(e))	**180**	(4) **DEFG**	→	↼↽	0	$\Theta(2^n)$
Divergent (fig. 2.8(f))	**134**	(4) **BDG**	↼↽	←	·	∞
	142	(2) **BG**	←	←	·	
	150	(1) **BDEG**	↼↽	↼↽	·	
	156	(2) **EG**	·	↼↽	·	

Table 2.2 – Classification des comportements totalement asynchrones ($\alpha = 0$) des DQECA.

La principale difficulté pour prouver ces résultats est de mettre en place un formalisme qui permette de s'assurer que les preuves soient correctes et qu'aucun cas ne soit oublié. Nous verrons que ce point est critique car il est en effet souvent facile de se faire une intuition assez précise de ce qu'il se passe, mais la preuve passe souvent par des énumérations de cas difficile à contrôler où surgisse souvent des problèmes, ou des conjonctions de phénomènes, qui n'avaient pas pu être observés lors des simulations. C'est la raison pour laquelle nous avons introduit les notations des transitions actives qui permettent d'obtenir directement à la lecture de la règle les variations du variant considéré pour prouver la convergence. L'obtention et la vérification des preuves en sont grandement facilitées.



(a) Coupons (b) Monotone (c) Marche biaisée

(d) Marche aléatoire (e) Marche fuyante (f) Divergent

Figure 2.8 – Illustrations des comportements totalement asynchrones des DQECA.

Automates collectionneurs de coupons. Cette classe regroupe les DQECA représentés par **C** et **CF** qui ont pour seuls voisinages actifs les 0 ou les 1 isolés. La convergence est obtenue dès que toutes les cellules correspondantes ont été mises à jour ce qui prend un temps normalisé $\Theta(\log n)$ en espérance depuis la pire configuration, $(010)^{\lfloor n/3 \rfloor}1^a$ avec $a \in \{0, 1, 2\}$.

Automates monotones et marches aléatoires biaisées favorablement. Cette classe regroupe les DQECA représentés par **CDG**(**128**), **DG**(**132**), **CG**(**136**), **G**(**140**), **CDFG**(**160**), **DFG**(**164**), **CFG**(**168**) et **FG**(**172**) pour les monotones, et **BCDG**(**130**) et **BCDFG**(**162**) pour les marches aléatoires biaisées favorablement. Pour tous ces automates, chaque 1-region s'érode avec une probabilité $\geqslant 1/n$ à chaque pas de temps, et pour certains, les 1 isolés ou les 0 sont éliminés. On introduit donc la fonction potentiel $\Phi(c) = |c|_1 + Z(c)$. Quelque soit la configuration c, on a $0 \leqslant \Phi(c) \leqslant 3n/2$. De plus, si $\Phi(c) = 0$, alors c est stable. Posons $\Phi_t = \Phi(c^t)$ tant c^t n'est pas une configuration stable ; lorsque c^t est stable pour la première fois, si $\Phi_t \neq 0$, nous prolongeons ce processus artificiellement jusqu'à ce qu'il s'annule, en en faisant diminuer de 1 la valeur avec probabilité $1/n$ et en le laissant inchangé sinon. On peut dire que (Φ_t) est une sorte de *processus zombie* qui survit au-delà de la convergence de c^t en perpétuant les mêmes mouvements. Ceci permet d'assurer que la variable aléatoire $T = \min\{t : \Phi_t = 0\}$ domine stochastiquement le temps de convergence de c^t. On démontre alors en sommant les contributions de chacun des types de cellules actives de la configuration à la variation de la fonction potentiel (v. le cas de la règle **BCFG**(**170**) ci-après) que, tant que c^t n'est pas une configuration stable, $\mathbb{E}[\Delta\Phi_{t+1}|c^t] \leqslant -1/n$. Φ_t vérifie donc les conditions du lemme 2.19 avec $\Phi_0 \leqslant 3n/2$ et $\varepsilon = 1/n$. L'espérance temps de relaxation normalisé est donc $\leqslant \frac{1}{n} \cdot \frac{3n/2}{1/n} = \Theta(n)$. Ce temps est atteint depuis la configuration initiale $1^{n-2}0^2$ pour tous ces automates.

Automates marches aléatoires non-biaisées. Cette classe regroupe les DQECA représentés par **BCG**(**138**), **BCDEG**(**146**), **CEFG**(**184**), **BCFG**(**170**), **BCDEFG**(**178**) et **BCD**(**194**). Il s'agit de coupler ces processus avec un processus de type I ou II. Voyons par exemple, le cas du shift **BCFG**(**170**). Nous considérons la fonction potentiel $\Phi(c) = |c|_1$ et posons $\Phi_t = \Phi(c^t)$. Φ_t prend ses valeurs dans $\{0, \dots, n\}$ et si $\Phi_t = 0$ ou n, c^t est une configuration stable. L'automate **BCFG**(**170**) admet quatre voisinages actifs : **B** (001) et **F** (101) qui augmentent de un le nombre de 1, et **C** (010) et **G** (110) qui diminuent de un le nombre de 1. Ainsi,

$$\mathbb{E}[\Delta\Phi_{t+1}|c^t] = \frac{1}{n}(|c^t|_{001} + |c^t|_{101} - |c^t|_{010} - |c^t|_{110}) = \frac{1}{n}(|c^t|_{01} - |c^t|_{10}) = 0,$$

c.-à-d. Φ_t est bien une martingale. De plus, tant que la configuration n'est pas stable :

$$\Pr\{\Delta\Phi_{t+1} \geqslant 1|c^t\} = \Pr\{\Delta\Phi_{t+1} \leqslant -1|c^t\} = \frac{1}{n}(|c^t|_{001} + |c^t|_{101}) = \frac{1}{n}|c^t|_{01} \geqslant \frac{1}{n}$$

Ainsi (Φ_t) est bien un processus de type I dont l'espérance du temps d'atteinte de 0 ou n est $O(n^3)$ (lemme 2.21). Cette borne est atteinte en prenant pour configuration initiale $0^{\lfloor n/2 \rfloor}1^{\lceil n/2 \rceil}$. Le temps de relaxation normalisé est donc bien $\Theta(n^2)$ pour le shift. La preuve est similaire pour les autres automates de cette classe, avec quelques raffinements pour les automates **BCDEG**(**146**) et **BCDEFG**(**178**), v. [111].

Automates marches aléatoires fuyantes. Cette classe regroupe les DQECA représentés par **DEFG**(**180**), pour lequel les 1-régions s'érodent par la gauche et suivent des marches aléatoires à droite, peuvent fusionner, mais la dernière ne peut disparaître. Ainsi, dès que la configuration initiale est non-triviale, les 1-régions vont commencer par fusionner, puis la configuration va évoluer aléatoirement jusqu'à ce que l'unique 1-région survivante finisse par recouvrir totalement la configuration au bout d'un temps exponentiel.

Commençons par étudier la dernière phase : le cas d'une configuration avec une unique 1-région de longueur $1 \leqslant k \leqslant n-1$. Considérons la fonction potentiel $\Phi(c) = |c|_0$ et posons $\Phi_t = \Phi(c^t)$. Les voisinages actifs de **DEFG**(**180**) sont : **D** (011) et **G** (110) qui augmentent de un le nombre de 0, et **E** (100) et **F** (101) qui diminuent de un le nombre de 0. Ainsi,

- Si $1 \leqslant \Phi_t \leqslant n-2$, alors la configuration compte une cellule active de type **D** ($+1$ de 0), une de type **G** ($+1$) et une de type **F** (-1) donc : $\Delta\Phi_{t+1} = +1$ avec probabilité $2/n$, $\Delta\Phi_{t+1} = -1$ avec probabilité $1/n$, et $\Delta\Phi_{t+1} = 0$ sinon ;
- si $\Phi_t = n-1$, alors la configuration compte une seule cellule active, de type **C** (-1 de 0), donc $\Delta\Phi_{t+1} = -1$ avec probabilité $1/n$, et $\Delta\Phi_{t+1} = 0$ sinon ;
- enfin, si $\Phi_t = 0$, la configuration est stable et le processus a convergé.

Posons $T_k = \mathbb{E}[\min\{t : \Phi_t = 0\}]$ où k désigne la longueur initiale de l'unique 0-région. Le processus étant markovien, nous avons :

$$\begin{cases} T_{n-1} = 1 + \dfrac{1}{n}T_{n-2} + \left(1 - \dfrac{1}{n}\right)T_{n-1} \\ T_k = 1 + \dfrac{1}{n}T_{k-1} + \left(1 - \dfrac{3}{n}\right)T_k + \dfrac{2}{n}T_{k+1}, \quad \text{pour } 1 \leqslant k \leqslant n-2 \\ T_0 = 0. \end{cases}$$

En prenant l'espérance de ces équations, puis en résolvant le système, nous en déduisons que $T_k = n2^n(1-2^{-k}) - kn$ pour tout $1 \leqslant k \leqslant n-1$. Le pire temps de convergence normalisé, $\hat{T}_{n-1}$, est donc équivalent à 2^n.

Étudions à présent le cas général d'une configuration initiale c non-triviale quelconque. Nous allons démontrer que le temps de convergence est stochastiquement dominé par celui partant d'une unique `1`-région par un argument de couplage probabiliste. Pour cela marquons une des `1`-régions de c et construisons une configuration γ consistant en une unique `1`-région alignée avec la région marquée de c et de même longueur. On considère alors le couplage suivant : les cellules mises à jour à chaque pas de temps sont les mêmes dans les deux configurations. Après chaque transition, le bord droit de la `1`-région de γ^t est réalignée avec le bord droit de la `1`-région marquée de c^t. Si la `1`-région marquée de c^t fusionne avec une autre `1`-région, la `1`-région issue de cette fusion devient celle marquée. Remarquons que comme à chaque pas de temps la cellule mise à jour est tirée uniformément parmi toutes les cellules, le réalignement ne modifie en rien la loi d'évolution ni de γ^t ni de c^t. Nous allons démontrer qu'après chaque réalignement, l'unique `1`-région de γ^t est toujours incluse dans la `1`-région marquée de c^t. C'est vrai par construction à $t = 0$. Maintenant, supposons que la `1`-région de γ^t soit incluse dans et alignée sur le bord droit de la `1`-région marquée de c^t. Considérons le voisinage de la cellule mise à jour dans γ^t. Les seuls cas à considérer sont ceux qui touchent l'unique `1`-région de γ^t ou la `1`-région marquée de c^t.

- Si le voisinage mis à jour est **G** ou **E** : la transition a lieu sur le bord droit des deux `1`-régions et l'évolution de la `1`-région marquée de c^t est la même sauf si elle fusionne avec une autre, ce qui ne peut que l'agrandir.
- Si le voisinage est **D** : la transition a lieu sur le bord gauche de la `1`-région de γ^t qui rétrécit de 1 et la `1`-région de c^t est soit inchangée soit évolue de la même façon.
- Si le voisinage est **C** : soit la `1`-région marquée de c^t est aussi de longueur 1 et reste inchangée, soit elle est de longueur $\geqslant 2$ et sa longueur diminuant d'au plus 1, la `1`-région de γ^t tiendra bien dedans une fois le réalignement effectué.
- Enfin, si le voisinage est **F** : soit $c^t = \mathtt{1}^n$ a déjà convergé, soit $c^t = \gamma^t$ et les deux configurations évoluent de façon identique.

Il s'en suit que la convergence vers $\mathtt{1}^n$ de c^t est dominée stochastiquement par celle de γ^t dont l'espérance du temps de convergence normalisé est borné par 2^n comme nous l'avons prouvé ci-dessus. Ainsi, le temps de relaxation normalisé de l'automate **DEFG**(**180**) est $\Theta(2^n)$.

Notons que cet automate est le seul DQECA à présenter un *comportement métastable*, puisque son comportement *observable* (après une phase très courte de fusions des `1`-régions, une petite `1`-région dont la taille oscille se déplace vers la droite) est radicalement différent du comportement asympotique (configuration toute-noire).

Remarque 2.26. Notre preuve par couplage ne nous permet pas d'obtenir de bornes sur la durée de la phase de fusions des `1`-régions, le seul phénomène observable par simulation de cet automate. Il faut reconnaître que bien qu'il soit facile d'intuiter ce qu'il se passe, prouver une telle borne se révèle être assez délicat puisque les `1`-régions ne se déplacent pas comme des particules individuelles mais plutôt "comme des poules" : la frontière droite (la tête de la poule) suit une marche aléatoire non-biaisée qui est poussée par la frontière gauche (le corps de la poule) qui suit une marche aléatoire biaisée vers la droite. La conséquence en est que la distance entre deux `1`-régions voisines croît en espérance paradoxalement à moins que les

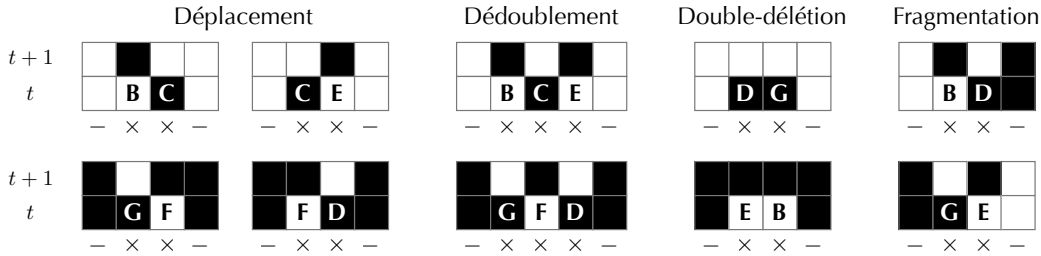


Figure 2.9 – Nouvelles évolutions locales des régions pour les DQECA en régime α-asynchrone. (Les cellules mises à jour au temps t sont marquées d'une croix $\times$)

deux 1-régions soient de longueur 1, il s'en suit que ces régions ne se rapprochent pas à un taux 1 mais plutôt à un taux inférieur à $1/2$ comme en témoigne les pentes des trajectoires de ces régions sur les figures 2.6 et 2.8(e). L'analyse précise de la phase de fusion se révèle donc bien plus délicate qu'il n'y paraît. C'est dans ces subtilités de comportements qu'on mesure à quel point l'analyse de ces automates peut se révéler rapidement difficile.

Automates divergents. Cette classe regroupe les DQECA représentés par **BDG**(**134**), **BG**(**142**), **BDEG**(**150**) et **EG**(**156**), pour lesquels les longueurs des 0- et des 1-régions suivent des marches aléatoires non-biaisées et pour lesquels aucune région ne peut fusionner avec ses voisines puisque les transitions **C** et **F** sont inactives. Il s'en suit que toute configuration non triviale et différente de $(01)^{n/2}$ (lorsque n est pair) évolue perpétuellement, d'où un temps de relaxation normalisé infini.

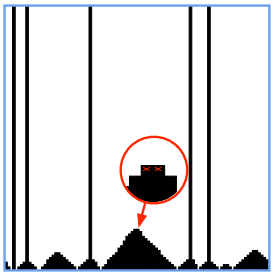
Nous renvoyons le lecteur à l'article [111] pour la preuve complète du théorème 2.25. Nous avons ainsi démontré l'utilité de la notion par lettre des automates qui permet d'obtenir et de vérifier facilement l'évolution du variant à partir du code de l'automate en régime totalement asynchrone. Étudions à présent le comportement α-asynchrone.

3.5 Étude du régime α-asynchrone

En dynamique α-asynchrone, toute combinaison de cellules actives voisines a toujours une probabilité non-nulle de se mettre à jour simultanément. Ceci engendre de nouvelles évolutions locales possibles pour les régions (v. figure 2.9) :

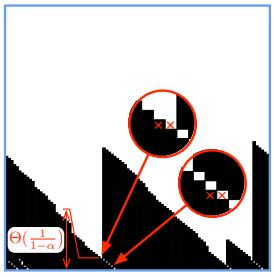
- le *déplacement* d'une cellule isolée : noire vers la gauche si **B** et **C** sont actifs, ou vers la droite si **C** et **E** sont actifs ; blanche vers la gauche si **F** et **G** sont actifs, ou vers la droite si **D** et **F** sont actifs.
- le *dédoublement* d'une cellule isolée : noire si **B**, **C** et **E** sont actifs, ou blanche si **D**, **F** et **G** sont actifs.
- la *double-délétion* : élimination d'une région de longueur 2, noire si **D** et **G** sont actifs, ou blanche si **B** et **E** sont actifs.
- la *fragmentation* d'une frontière : 01 si **B** et **D** sont actifs, ou 10 si **E** et **G** sont actifs.

Les influences de ces nouvelles évolutions locales les unes sur les autres sont étonnamment intriquées et l'activité ou non d'un voisinage peut conduire à des comportements radicalement différents qui peuvent se révéler très difficiles à analyser. Par exemple, les automates **BCDFG**(**162**) et **BCDEFG**(**178**) diffèrent uniquement sur l'activité ou non du voisinage



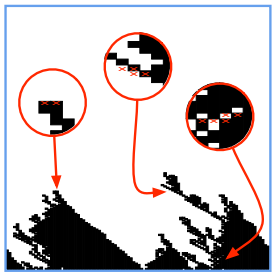
Règle **DG(132)**
$p_{\text{init}} = 0.9$ et $\alpha = 0.9$

(a) Monotone



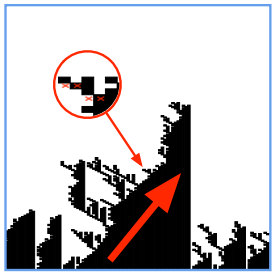
Règle **CFG(168)**
$p_{\text{init}} = 0.98$ et $\alpha = 0.8$

(b) Monotone à amorce



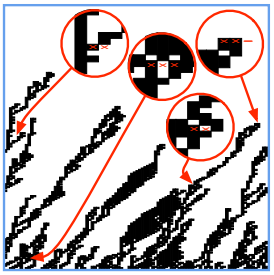
Règle **BCDFG(162)**
$p_{\text{init}} = 0.9$ et $\alpha = 0.5$

(c) Marche biaisée

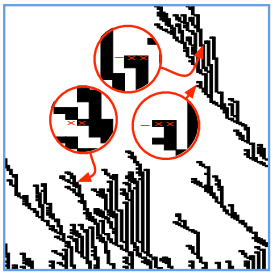


Règle **BCD(194)**
$p_{\text{init}} = 0.9$ et $\alpha = 0.5$

(d) Marche aléatoire biaisée par les fragmentations

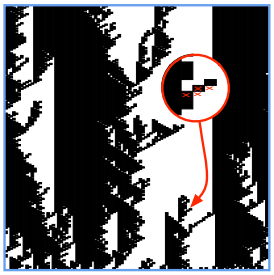


Règle **DEFG(180)**
$p_{\text{init}} = 0.5$ et $\alpha = 0.5$



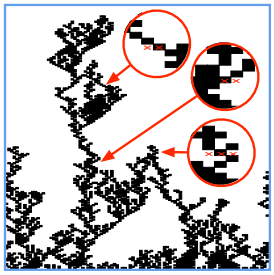
Règle **BDG(134)**
$p_{\text{init}} = 0.2$ et $\alpha = 0.5$

(e) Marche aléatoire (biaisée ?) avec fragmentation négligeable

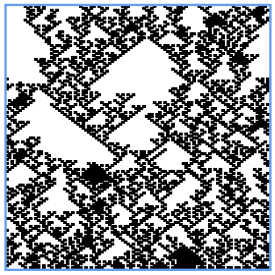


Règle **CEFG(184)**
$p_{\text{init}} = 0.5$ et $\alpha = 0.5$

(f) Marche aléatoire

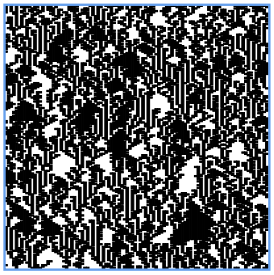


Règle **BCDEG(146)**
$p_{\text{init}} = 0.5$ et $\alpha = 0.5 < \alpha_c^{\mathbf{146}}$



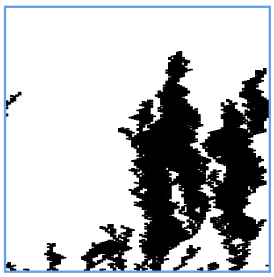
Règle **BCDEG(146)**
$p_{\text{init}} = 0.5$ et $\alpha = 0.75 > \alpha_c^{\mathbf{146}}$

(g) Transition de phase polynomial/exponentiel à $\alpha_c^{\mathbf{146}} \approx 0.66$

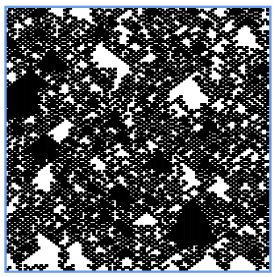


Règle **BDEG(150)**
$p_{\text{init}} = 0.5$ et $\alpha = 0.5$

(h) Convergent non-structuré



Règle **BCDEFG(178)**
$p_{\text{init}} = 0.2$ et $\alpha = 0.25 < \alpha_c^{\mathbf{178}}$



Règle **BCDEFG(178)**
$p_{\text{init}} = 0.2$ et $\alpha = 0.75 > \alpha_c^{\mathbf{178}}$

(i) Transition de phase polynomial/exponentiel à $\alpha_c^{\mathbf{178}} \approx 0.50$

Figure 2.10 – Illustrations des comportements α-asynchrones des DQECA.

E. Pourtant, alors que le comportement α-asynchrone de **BCDFG**(**162**) est relativement simple à analyser (v. illustration figure 2.10(c)), celui de l'automate **BCDEFG**(**178**) présente dans les simulations une transition de phase sur le temps de convergence polynomial/exponentiel autour d'une valeur critique du taux d'asynchronisme $\alpha_c^{\mathbf{178}} \approx 0.50$ (v. illustration figure 2.10(i)). De part l'intrication de ces différentes évolutions possibles, l'analyse de ce dernier soulève de nombreuses difficultés. Nous ne disposons actuellement que d'une borne très lâche sur cette valeur critique, $\alpha_c^{\mathbf{178}} < 0.996$ pour les configurations infinies de support borné, obtenue par Damien Regnault [252] en procédant par couplage avec des processus de percolation dirigée. Nous proposons également de satisfaire une curiosité naturelle avec la figure 2.11, qui illustre l'évolution des comportements au fur et à mesure que l'on active des voisinages. Il est assez surprenant de constater l'intrication des différentes classes de comportements : aucun cluster de comportements similaires ne semble vraiment émerger de cette figure (quelqu'en soit la position des nœuds), en particulier les différentes classes ne semblent pas correspondre à la constitution d'une multi-coupe minimale.

Le fait que plusieurs cellules puissent se mettre à jour simultanément complique également considérablement l'espace des possibles. La prise en compte des différents phénomènes requiert une description plus fine que précédemment des fonctions potentiel. Regarder les voisinages actifs ne suffit plus et il faut augmenter l'étendue des voisinages analysés. Certaines combinaisons locales jouent en effet un rôle déterminant dans la convergence de l'automate : p. ex., la fragmentation accélère considérablement la convergence de l'automate **BCD**(**194**) en régime α-asynchrone par rapport au régime totalement asynchrone (v. fig. 2.10(d)).

Il convient donc de pondérer les voisinages clés différemment de ceux dont l'influence est moindre. Il se trouve que le nombre de cas à étudier explose et le risque devient très grand d'oublier des cas à la fois dans la preuve et lors de la recherche de la fonction potentiel. Ceci est une source dangeureuse de "bugs" et de perte de temps dans les preuves. Aussi, nous avons proposé un formalisme spécifique pour ces analyses : les *arbres de masques*. Cette structure permet à la fois de trouver puis d'encoder une fonction potentiel de manière compacte sans omettre de cas et surtout d'en certifier l'analyse des variations de manière quasi-automatique. Nous verrons que Damien Regnault a su dans sa thèse pousser ce formalisme très loin pour obtenir et analyser grâce à lui des fonctions potentiel fort sophistiquées pondérant judicieusement des phénomènes contradictoires complexes et démontrer une première borne sur la transition de phase de l'un des automates au comportement le plus évolué, **BDG**(**134**) (v. [251, 254]).

Définition 2.27 (Masque de voisinage). *Un* masque $\dot{m}$ *est un mot sur* $\{0, 1, \dot{0}, \dot{1}\}$ *contenant exactement une lettre pointée,* $\dot{0}$ *ou* $\dot{1}$. *On dira que la cellule* i *d'une configuration* c correspond au masque $\dot{m} = m_{-k} \dots \dot{m}_0 \dots m_\ell$ *si* $c_{i-k} \dots c_i \dots c_{i+\ell} = m_{-k} \dots m_0 \dots m_\ell$. *Nous désignons par* m *le mot* $m_{-k} \dots m_0 \dots m_\ell$ *dont on a ôté le point.*

Remarquons que le nombre de cellules correspondant à un masque $\dot{m}$ donné est exactement le nombre d'occurrences du motif m dans la configuration : $|c|_m$.

Définition 2.28 (Base de masques). *Une* base de masques $\mathcal{B}$ *est un ensemble fini de masques tel que pour toute configuration* c *et toute cellule* i, *il existe exactement un unique masque* $\dot{m} \in \mathcal{B}$ *qui corresponde à la cellule* i.

Disposer d'une base de masques garantit que tous les cas sont pris en compte une et une seule fois. La définition suivante permet de construire aisément de telles bases de masques.

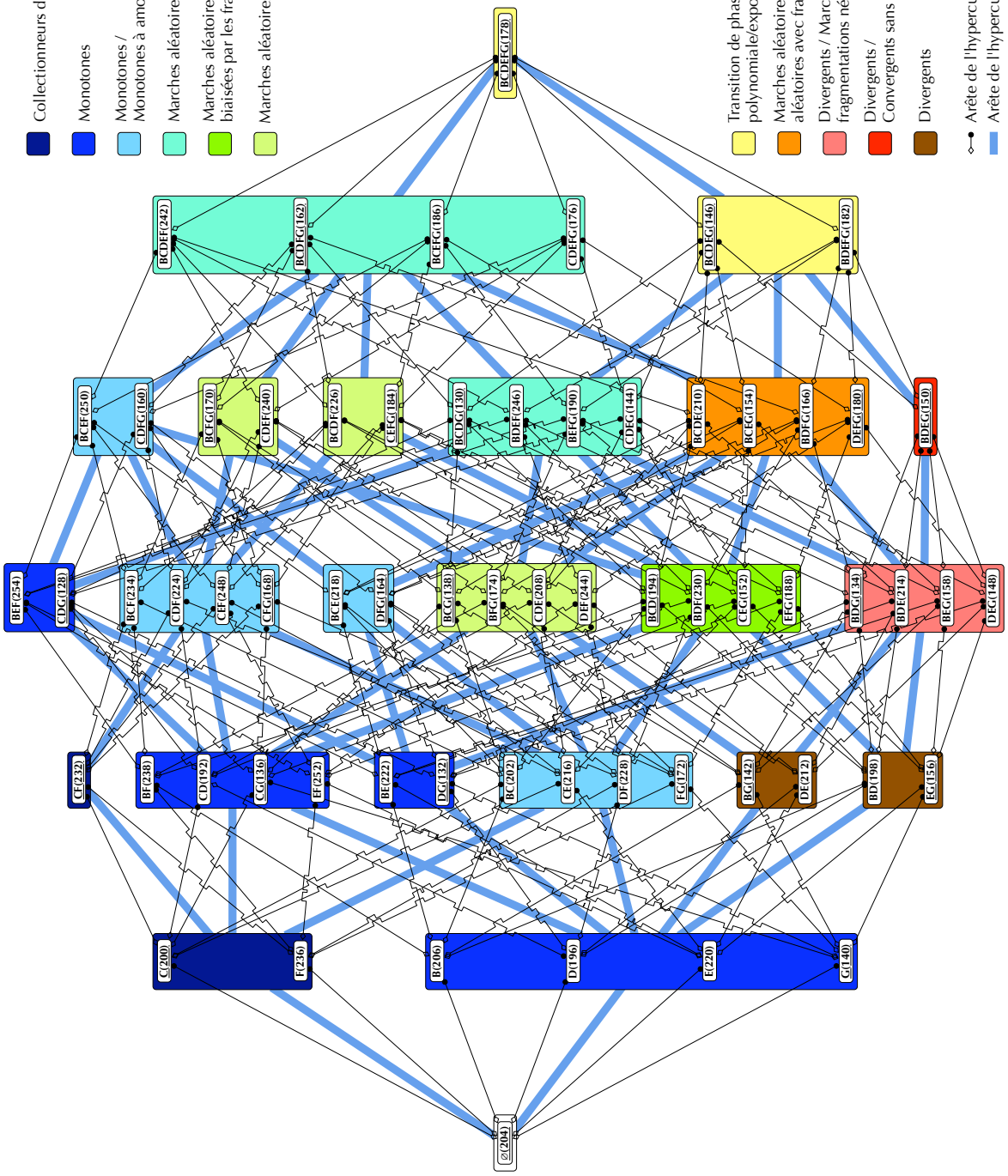


Figure 2.11 – Hypercube des noms des DQECA regroupés par symétries et comportements.

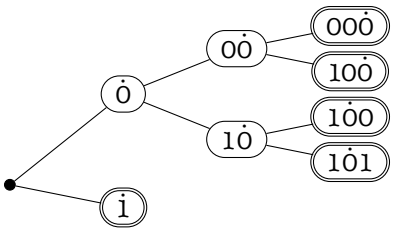


Figure 2.12 – Un arbre de masques $\mathcal{T}$ engendrant $\mathcal{B}_\mathcal{T} = \{000̇, 100̇, 1̇00, 1̇01, 1̇\}$, sa base de masques associée. Pour une plus grande lisibilité, les feuilles de l'arbre sont entourées d'un double-trait.

Définition 2.29 (Arbre de masques). *Un* arbre de masques $\mathcal{T}$ *est un arbre binaire enraciné et étiqueté. La racine n'a pas d'étiquette. Les deux fils de la racines sont étiquetés* 0̇ *et* 1̇. *Les fils d'un nœud v d'étiquette $\dot{m}_v$ reçoivent pour étiquettes soit* $0\dot{m}_v$ *et* $1\dot{m}_v$, *soit* $\dot{m}_v0$ *et* $\dot{m}_v1$. *On note $\mathcal{B}_\mathcal{T}$ l'ensemble des étiquettes des feuilles de l'arbre de masque $\mathcal{T}$.*

Théorème 2.30 ([112], [254]). *Un ensemble de masques $\mathcal{B}$ est une* base de masques *si et seulement s'il existe un arbre de masques $\mathcal{T}$ tel que $\mathcal{B} = \mathcal{B}_\mathcal{T}$.*

On dit que $\mathcal{B}_\mathcal{T}$ est la *base de masques engendrée par* $\mathcal{T}$. La preuve de ce théorème procède par induction sur la taille de l'arbre pour le "si" et sur la taille de la base considérée pour le "seulement si" (v. [254]). Le corollaire suivant sera très utile pour obtenir des identités de comptage de motifs dans les configurations.

Corollaire 2.31. *Étant donné un nœud v d'étiquette $\dot{m}$ dans un arbre de masques $\mathcal{T}$, si l'on note $\mathcal{B}_v$ l'ensemble des étiquettes des feuilles descendantes de v, alors pour toute configuration c et toute cellule i correspondant au masque $\dot{m}$, il existe un unique masque $\dot{p} \in \mathcal{B}_v$ qui corresponde à la cellule i. En particulier,*

$$|c|_m = \sum_{\dot{p}\in\mathcal{B}_v} |c|_p$$

Par exemple, le sous-arbre enraciné en 0̇ dans l'arbre de masques de la figure 2.12, démontre l'identité : $(\forall c)$, $|c|_0 = |c|_{000} + 2|c|_{100} + |c|_{101}$.

Définition 2.32 (Fonction potentiel de poids locaux). *Une* fonction potentiel de poids locaux Φ *est définie par un arbre de masques pondéré $(\mathcal{T}, \pi)$ muni de poids sur les feuilles. Φ associe à chaque configuration c, le potentiel :*

$$\Phi(c) = \sum_i \Phi_i(c),$$

où $\Phi_i(c)$ est le potentiel attribué à la cellule i de c, c.-à-d. $\Phi_i(c) = \pi_{\dot{m}}$ où $\dot{m}$ est l'unique masque de $\mathcal{B}_\mathcal{T}$ correspondant à la cellule i de c et $\pi_{\dot{m}}$ désigne le poids de la feuille d'étiquette $\dot{m}$. On notera $\mathcal{B}_\Phi$ la base de masques engendrée par l'arbre définissant Φ.

La figure 2.13 présente une fonction potentiel de poids locaux, qui sera utilisée par la suite pour analyser l'automate **BCD**(**194**) page 107.

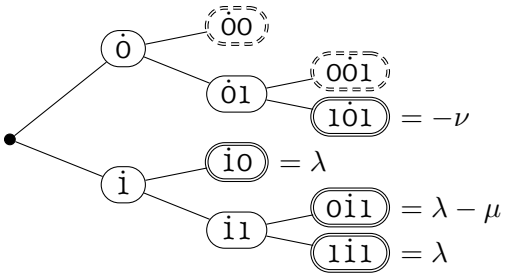


Figure 2.13 – Une fonction potentiel de poids locaux définie par son arbre de masques pondéré : $\Phi(c) = \lambda|c|_{1} - \mu|c|_{011} - \nu|c|_{101}$ de base $\mathcal{B}_\Phi = \{\dot{0}0, 0\dot{0}1, 1\dot{0}1, \dot{i}0, 0\dot{i}1, 1\dot{i}1\}$. Les feuilles de poids nul seront toujours représentées en pointillés.

Fait 2.33. *Par construction,* $\Phi(c) = \sum_{\dot{m}\in\mathcal{B}_\mathcal{T}} |c|_m \pi_{\dot{m}}$.

Une fonction potentiel de poids locaux donne à chaque cellule un poids en fonction de son voisinage étendu. Le poids de chaque cellule est déterminé par le masque auquel elle correspond dans la base de masques engendrée par l'arbre associé. En ce qui nous concerne, le poids d'une cellule sera d'autant plus élevé que sa mise-à-jour dans ce voisinage étendu est favorable à la convergence de l'automate.

Les variations d'une fonction potentiel de poids locaux s'analysent en construisant un arbre de masques suffisamment profond pour que l'on puisse déterminer l'évolution du motif de chaque feuille et donc l'espérance de la variation de la fonction potentiel sur ce motif. Cet arbre de preuves a souvent une structure similaire à celui definissant la fonction potentiel. Nous verrons ceci en détail par la suite.

Le théorème et les conjectures suivantes résument l'état de nos connaissances sur les régimes α-asynchrones des DQECA. On peut tout de suite remarquer que la situation est singulièrement plus compliquée que dans le cas totalement asynchrone.

Théorème 2.34 ([112], [254]). *En régime α-asynchrone avec $0 < \alpha < 1$, parmi les soixante-quatre DQECA,*

- *∅(**204**) est inerte et son temps de relaxation normalisé ($\widehat{\mathrm{TR}}$) est nul ;*
- *trois, représentés par **C**(**200**) et **CF**(**232**), se comportent comme des collectionneurs de coupons, leur $\widehat{\mathrm{TR}}$ est $\Theta(-\frac{\alpha \log n}{\log(1-\alpha)})$ pour deux d'entre eux, et compris entre $O(\frac{\log n}{1-\alpha})$ et $\Omega(-\frac{\alpha \log n}{\log(1-\alpha)})$ pour l'autre ;*
- *douze, représentés par **CDG**(**128**), **DG**(**132**), **CG**(**136**) et **G**(**140**), se comportent de façon monotone et leur $\widehat{\mathrm{TR}}$ est $\Theta(n)$, indépendant de α.*
- *douze, représentés par **CDFG**(**160**), **DFG**(**164**), **CFG**(**168**) et **FG**(**172**), se comportent de façon monotone après une phase d'initialisation qui amorce le comportement monotone, et leur $\widehat{\mathrm{TR}}$ est $\Theta(\frac{1}{1-\alpha} + n)$ pour huit d'entre eux et compris entre $O(\frac{1}{1-\alpha} + n)$ et $\Omega(n)$ pour les quatre autres ;*
- *huit, représentés par **BCDG**(**130**) et **BCDFG**(**162**), se comportent comme des marches aléatoires biaisées et leur $\widehat{\mathrm{TR}}$ est compris entre $O(\frac{n}{1-\alpha})$ et $\Omega(n)$;*
- *quatre, représentés par **BCD**(**194**), se comportent comme des marches aléatoires biaisées par les fragmentations et leur $\widehat{\mathrm{TR}}$ est $\Theta(\frac{n}{\alpha(1-\alpha)})$;*

- *huit, représentés par* **BCG**(**138**)*,* **BCFG**(**170**) *et* **CEFG**(**184**)*, se comportent comme des marches aléatoires non-biaisées et leur* $\widehat{\mathsf{TR}}$ *est compris entre* $O(\frac{n^2}{1-\alpha})$ *et* $\Omega(n)$ *;*
- *quatre, représentés par* **BG**(**142**) *et* **EG**(**156**)*, divergent* ($\widehat{\mathsf{TR}} = \infty$)*.*

Le comportement α*-asynchrone des douze derniers est encore largement non-déterminé et sujet aux conjectures énoncées page 118.*

Le tableau 2.3 présente en détail les résultats du théorème en les confrontant aux différents mécanismes présents dans chaque classe d'automates. Les dénominations de chaque classe de comportements sont inspirées des diagrammes espace-temps correspondant présentés à la figure 2.10. Le diagramme, figure 2.11, représente quant à lui l'enchevêtrement des comportements en fonction des différents voisinages activés dans chaque automate (*i.e.*, en fonction de leur code par lettres), qui comme nous l'avons dit précédemment ne semble pas correspondre à une partition naturelle de l'hypercube des noms des automates. En particulier, on peut très bien activer ou désactiver des voisinages de certains automates pour aller dans des classes aux comportements très différents ou non, plus complexes ou non.

Hormis le cas des collecteurs de coupons que nous ne ferons qu'évoquer ici, la preuve de ce théorème repose sur un formalisme issu des arbres de masques assurant que tous les cas sont correctement traités : les *arbres d'analyse*, que nous allons introduire sur un exemple simple, l'analyse des automates **G**(**140**) et **DG**(**132**) ci-après.

Les collectionneurs de coupons. Ces automates convergent dès qu'il n'y a plus de `1` ni/ou de `0` isolés. La différence avec l'analyse du régime totalement asynchrone réside dans le fait que pour les motifs alternés $\ldots \mathtt{1\dot{0}1\dot{0}1} \ldots$, il est nécessaire qu'un des voisins de la cellule pointée ne se mette pas-à-jour pour que la cellule devienne inactive à jamais, d'où l'apparition du facteur $(1-\alpha)$.

Les monotones. De même qu'en régime totalement asynchrone, le calcul de leur temps de relaxation est basé sur l'obtention d'une fonction potentiel décroissant strictement en espérance. L'analyse de ces automates ne présente pas de difficulté particulière et permettra donc de se familiariser avec notre formaliste sur un exemple tiré de cette classe, les automates **G**(**140**) et **DG**(**132**).

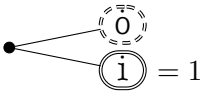
Les mises-à-jour des voisinages **G** (`110`) et **D** (`011`) rognent les `1`-régions de taille $\geqslant 2$ par la droite et par la gauche, respectivement. On introduit donc sans surprise la fonction potentiel $\Phi(c) = |c|_1$ qui compte le nombre de `1` dans la configuration, en attribuant un poids 0 au masque $\dot{\mathtt{0}}$ et 1 au masque $\dot{\mathtt{1}}$ (v. analyse 2.1(a) page 103). La preuve est basée sur l'analyse des variations locales de la fonction potentiel. Pour cela nous définissons une base de voisinages telle que nous sommes capables de déterminer localement l'espérance de l'évolution de Φ pour chacun d'eux : $\mathcal{B} = \{\dot{\mathtt{0}}, \mathtt{0\dot{1}0}, \mathtt{0\dot{1}1}, \mathtt{1\dot{1}}\}$ (v. analyse 2.1(b)). Considérons une configuration initiale arbitraire c. Posons $\Phi_t = \Phi(c^t)$ et $\Phi_{t,i} = \pi_{\dot{m}}$ où $\dot{m}$ est l'unique masque de $\mathcal{B}_\Phi$ correspondant à la cellule i de c^t. Par la linéarité de la valeur moyenne,

$$\mathbb{E}[\Delta\Phi_{t+1}] = \sum_i \mathbb{E}[\Delta\Phi_{t+1,i}]\,.$$

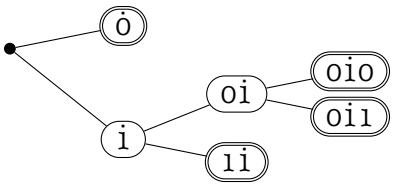
On omettra l'indice $t+1$ lorsque le contexte le permet. Nous présentons l'analyse de chacun des voisinages sous la forme de l'arbre d'analyse 2.1(c) page 103, véritable certificat du calcul de l'espérance des variations de Φ. La lecture de ce certificat se fait suivant les explications fournies par la figure (d) de l'analyse 2.1. Cet arbre d'analyse est obtenu en étendant les voisinages $\dot{\mathtt{0}}$ et $\dot{\mathtt{1}}$ jusqu'à ce qu'on puisse déterminer avec certitude les variations de Φ sur

Comportement $\alpha=0$	Comportement $0<\alpha<1$	DQECA (#)	Règle	←B D→ 01	←G E→ 10	0:F 1:C Dél.	←FG DF→ 0	←BC CE→ 1	0:DFG 1:BCE 2x	00:BE 11:DG 2x-Dél.	01:BD 10:EG Frag.	Temps de relaxation normalisé ($\widehat{\mathrm{TR}}$) $\alpha=0$	$0<\alpha<1$	$\alpha=1$
Inerte		**204**	(1) ∅	·	·	·	·	·	·	·	·		0	
Collectionneur de coupons		**200**	(2) **C**	·	·	~~1~~	·	·	·	·	·	$\Theta(\log n)$	$\Theta\left(-\frac{\alpha \log n}{\log(1-\alpha)}\right)$	1
		232	(1) **CF**	·	·	~~0~~,~~1~~	·	·	·	·	·		$O\left(\frac{\log n}{1-\alpha}\right), \Omega\left(-\frac{\log n}{\log(1-\alpha)}\right)$	∞
Monotone	Monotone	**128**	(2) **CDG**	→	←	~~1~~	·	·	·	~~11~~	·	$\Theta(n)$	$\Theta(n)$	
		132	(2) **DG**	→	←	·	·	·	·	~~11~~	·			
		136	(4) **CG**	·	←	~~1~~	·	·	·	·	·			
		140	(4) **G**	·	←	·	·	·	·	·	·			
	Monotone à amorce	**160**	(2) **CDFG**	→	←	~~0~~,~~1~~	↜z→	·	0	~~11~~	·		$O(\frac{1}{1-\alpha}+n), \Omega(n)$	
		164	(2) **DFG**	→	←	~~0~~	↜z→	·	0	~~11~~	·			
		168	(4) **CFG**	·	←	~~0~~,~~1~~	←	·	·	·	·		$\Theta(\frac{1}{1-\alpha}+n)$	
		172	(4) **FG**	·	←	~~0~~	←	·	·	·	·			
Marche aléatoire biaisée		**130**	(4) **BCDG**	↜z→	←	~~1~~	·	←	·	~~11~~	01		$O(\frac{n}{1-\alpha}), \Omega(n)$	
		162	(4) **BCDFG**	↜z→	←	~~0~~,~~1~~	↜z→	←	0	~~11~~	01			
Marche aléatoire non-biaisée	Marche biaisée par les fragmentations	**194**	(4) **BCD**	↜z→	·	~~1~~	·	←	·	·	01	$\Theta(n^2)$	$\Theta\left(\frac{n}{\alpha(1-\alpha)}\right)$	∞
		138	(4) **BCG**	←	←	~~1~~	·	←	·	·	·		$O(\frac{n^2}{1-\alpha}), \Omega(n)$	
		170	(2) **BCFG**	←	←	~~0~~,~~1~~	←	←	·	·	·			
		184	(2) **CEFG**	·	↜z→	~~0~~,~~1~~	←	→	·	·	10			
	Transition de phase : fragmentation négligeable/dominante	**146**	(2) **BCDEG**	↜z→	↜z→	~~1~~	·	↜z→	1	~~00~~,~~11~~	01,10		Transition de phase ? poly. pour $\alpha<\alpha_c^{146}$? exp. pour $\alpha>\alpha_c^{146}$?	
		178	(1) **BCDEFG**	↜z→	↜z→	~~0~~,~~1~~	↜z→	↜z→	0,1	~~00~~,~~11~~	01,10		Transition de phase ? polynomial pour $\alpha<\alpha_c^{178}$? exponentiel pour $\alpha>\alpha_c^{178}$? borne : $\alpha_c^{178}<0.996$ [252]	
Marche aléatoire fuyante	Marche aléatoire (biaisée ?) avec fragmentation négligeable	**180**	(4) **DEFG**	→	↜z→	~~0~~	↜z→	·	0	~~11~~	10	$\Theta(2^n)$	polynomial ?	
Divergent		**134**	(4) **BDG**	↜z→	←	·	·	·	·	~~11~~	01	∞	polynomial ? vrai pour $\alpha \geqslant 0.9999$ [251]	
	Convergent sans structure	**150**	(1) **BDEG**	↜z→	↜z→	·	·	·	·	~~00~~,~~11~~	01,10		$O(\alpha^n+(1-\alpha)^n)$, exponentiel ? (indépendant de α ?)	
	Divergent	**142**	(2) **BG**	←	←	·	·	·	·	·	·		∞	
		156	(2) **EG**	·	↜z→	·	·	·	·	·	10			

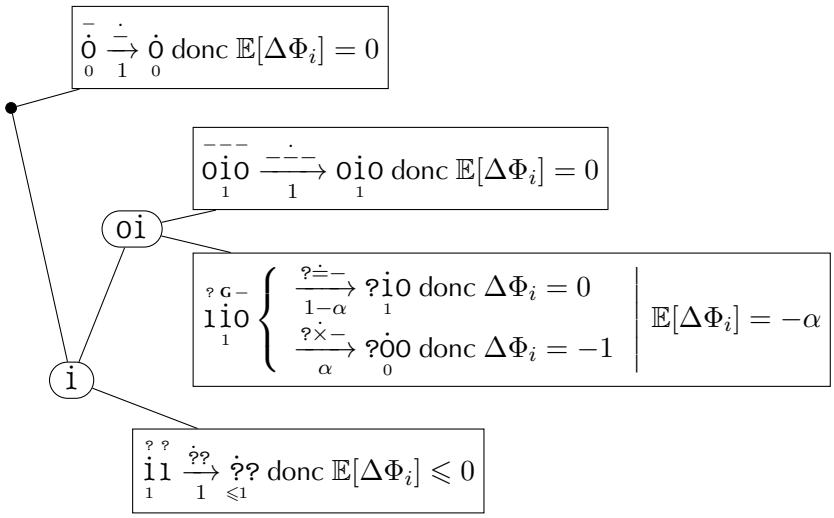
Table 2.3 – Classification des comportements des DQECA en régimes α-asynchrones.



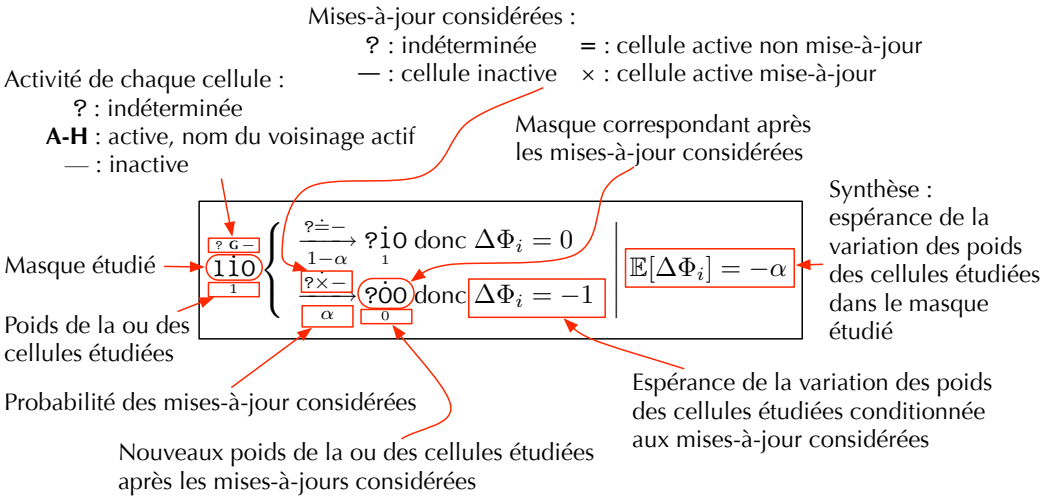
(a) Fonction potentiel pour l'analyse de **G**(**140**) et **DG**(**132**): $\Phi(c) = |c|_1$.



(b) Base de masques de l'analyse de **G**(**140**) et **DG**(**132**): $\mathcal{B} = \{\dot{\mathtt{o}}, \mathtt{oio}, \mathtt{oi1}, \mathtt{ii}\}$.



(c) Certificat du calcul des variations de Φ pour **G**(**140**) et **DG**(**132**).



(d) Schéma explicatif des notations utilisées. i désignera toujours l'indice de la cellule pointée du masque étudié.

Analyse 2.1: **G**(**140**) et **DG**(**132**)

chaque motif en fonction des différentes cellules mises-à-jour. L'espérance de la variation de Φ pour chacun des motifs est alors obtenu en prenant la somme des variations pondérée par la probabilité des mises-à-jour correspondantes.

Notre formalisme garantit qu'aucun cas n'a été oublié et permet, en faisant la somme des majorants obtenus pour chaque feuille pondérés par le nombre de cellules dans la configuration correspondant au motif associé, d'en déduire la majoration suivante.

Lemme 2.35 (G(140) et DG(132)). *Pour tout t, $\mathbb{E}[\Delta\Phi_{t+1}|c^t] \leqslant -\alpha|c^t|_{110}$.*

Ainsi, tant que la configuration contient le motif 11, la fonction potentiel décroît d'au moins α en espérance. Il s'en suit d'après le lemme 2.19 qu'au bout de $O(\frac{n}{\alpha})$ pas de temps ($O(n)$ en temps normalisé) en espérance, la configuration ne contient plus que des 1 isolés et est donc stable pour les règles **G**(**140**) ou **DG**(**132**). L'étude de la configuration 1^n0 démontre que le temps de relaxation normalisé est bien $\Theta(n)$ pour cet automate.

On pourra trouver ce formalisme excessif pour l'analyse d'un automate aussi simple, mais encore une fois, il s'agit ici uniquement d'un premier exemple pour se familiariser avec notre formalisme. Voyons à présent des cas plus intéressants.

Les monotones à amorces. Ce comportement n'était pas présent en régime totalement asynchrone. Pour ces automates, le pire temps de relaxation est obtenu en partant d'une configuration qui va créer avec probabilité constante un "germe" qui va se répandre au travers de toute la configuration. Étudions en détail le cas de l'automate **FG**(**172**).

Pour cet automate, les voisinages actifs sont **F** (101) et **G** (110). Les 0 isolés induisent un phénomène d'amorce. En effet, ces 0 isolés peuvent soit disparaître (mise-à-jour de lui seul, **F**), soit engendrer un motif 00 si leur voisine de gauche est active (mise-à-jour uniquement de leur voisine de gauche, **G**), ou, ce qui est nouveau par rapport au régime totalement asynchrone, se déplacer vers la gauche (mise-à-jour simultanée du 0 isolé et de sa voisine de gauche, **F** et **G**). La transition qui amorce le phénomène monotone est celle qui crée le motif stable 00 qui va ensuite progressivement se répandre dans la configuration. La différence avec le régime totalement asynchrone est que l'apparition de ce motif stable intervient au bout d'un temps qui dépend de α et non de n (dans le pire cas). Notons également que la phase de pré-amorce est plus complexe, puisque les 0 isolés se déplacent avant de disparaître ou d'engendrer le motif stable 00. Remarquons également que l'expansion des 0-régions est freinée par la fragmentation des frontières 10 dont il va falloir tenir compte.

Nous allons donc considérer les deux phases séparément. Considérons tout d'abord une configuration instable sans motif 00. Les 0 d'une telle configuration sont tous isolés et à moins que la configuration ne soit $(01)^{n/2}$, elle admet un motif 110. L'évolution de la configuration est résumée par la chaîne de Markov figure 2.14.

Si la configuration initiale est $(01)^{n/2}$, alors tous les 1 sont inactifs et avec probabilité $1-(1-\alpha)^{n/2}$, un des 0 est mis à jour pour donner un motif 1101 à moins qu'avec probabilité $\alpha^{n/2}$ tous les 0 soient mis-à-jour simultanément, mais alors la configuration devient stable. On peut donc à présent supposer que la configuration est instable, contient un motif 1101 mais

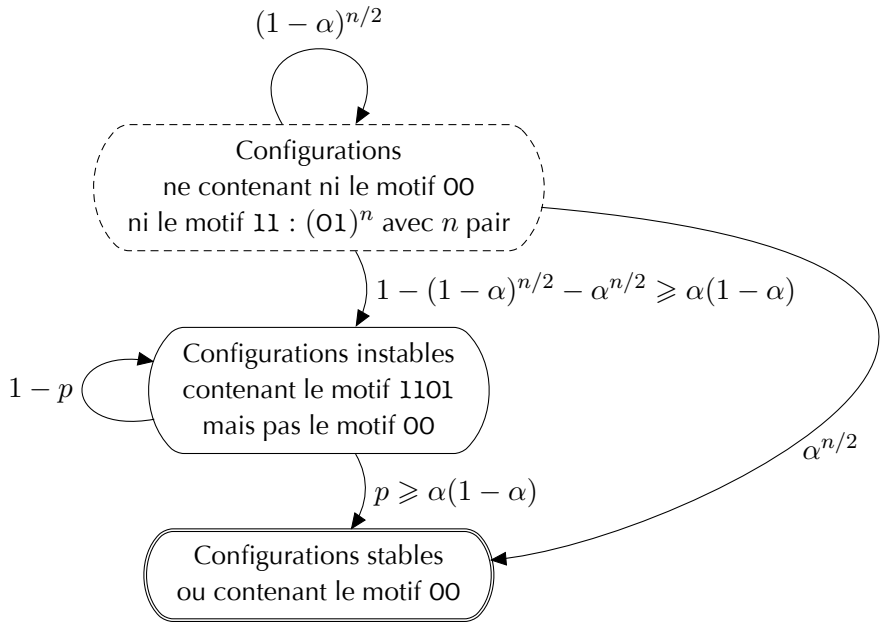


Figure 2.14 – Engendrement du motif 00 dans l'automate **FG**(**172**).

pas de motif 00. L'évolution du motif $11\dot{0}1$ est la suivante :

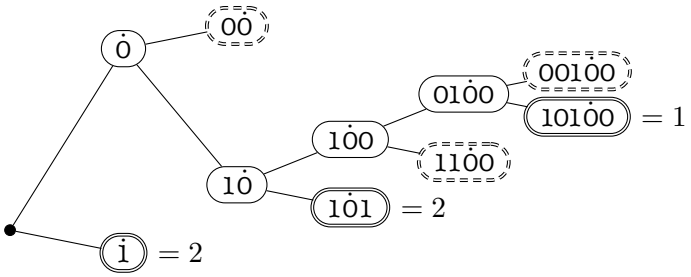
$$\overset{-\,\mathrm{G\,F}\,-}{11\dot{0}1} \begin{cases} \xrightarrow[(1-\alpha)^2]{-=\dot{=}-} 11\dot{0}1 \\ \xrightarrow[\alpha(1-\alpha)]{-\times\dot{=}-} 10\dot{0}1 \\ \xrightarrow[\alpha(1-\alpha)]{-=\dot{\times}-} 11\dot{1}1 \\ \xrightarrow[\alpha^2]{-\times\dot{\times}-} 10\dot{1}1 \end{cases}$$

Il s'en suit qu'avec probabilité $\geqslant \alpha(1-\alpha)$ ce motif engendre un motif 00 et que sinon, soit tous les 0 ont disparu et la configuration est stable, soit il existe toujours un motif 1101. On en conclut qu'au bout de $O(\frac{1}{\alpha(1-\alpha)})$ pas, c.-à-d. au bout d'un temps normalisé $O(\frac{1}{1-\alpha})$, soit la configuration est stable, soit elle contient un motif 00.

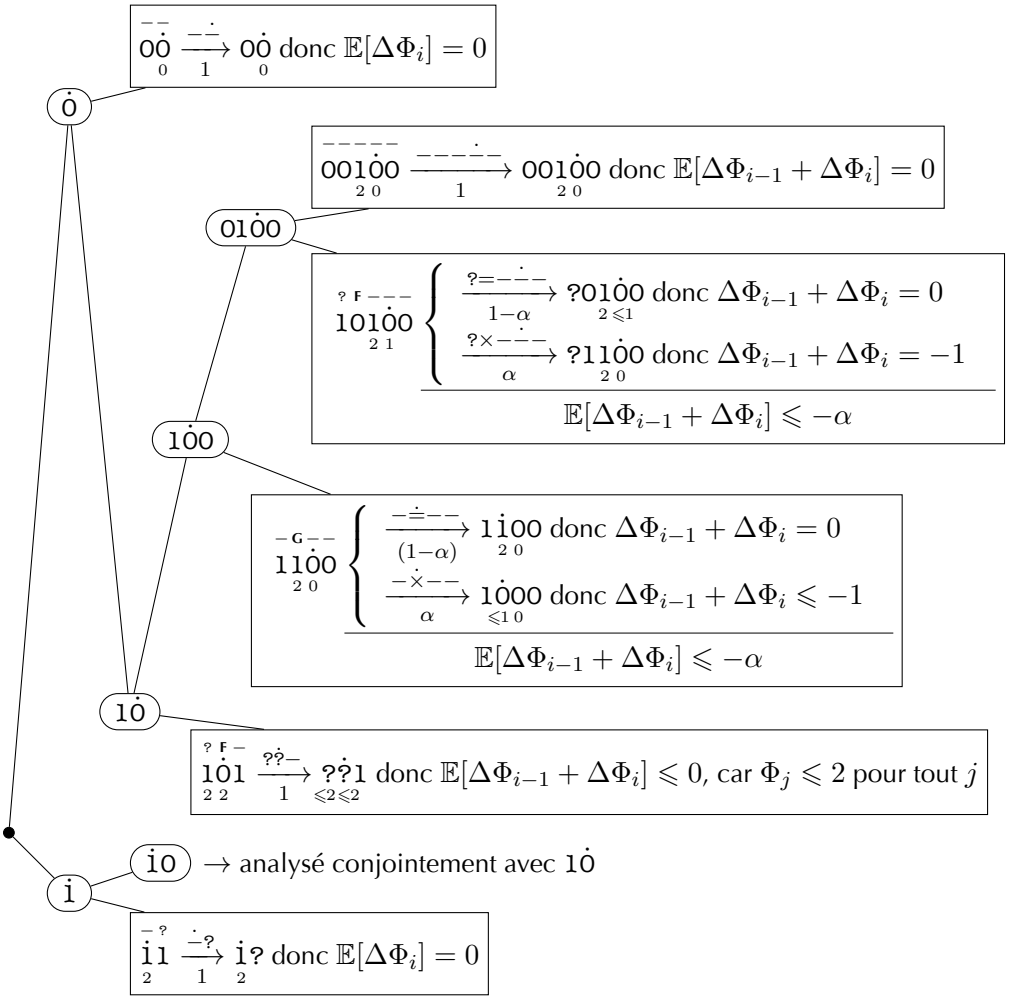
Supposons à présent que la configuration c contient un motif 00. Le comportement de cet automate semble clair : les 00, stables, rongent progressivement les 1-régions par la droite. Cette croissance est toutefois freinée par d'éventuels 0 isolés. En effet, pour que la 0-région de droite dans le motif $10\dot{1}00$ s'étende vers la gauche, il faut commencer par éliminer le 0 actif de gauche pour obtenir le voisinage $11\dot{1}00$ où $\dot{1}$ est actif. Les 0 du motif 10100 doivent donc être pénalisés car ce motif impose un ralentissement de la croissance des 0-régions.

L'approche développée par Damien Regnault dans sa thèse [254] procède alors par pondération des voisinages étendus en fonction de l'importance de leur mise-à-jour pour la convergence de la configuration. Nous choisissons donc la fonction potentiel Φ définie par l'arbre de l'analyse 2.2(a) qui pondère à 1 le masque $101\dot{0}0$, à 2 les masques $\dot{1}$ et $1\dot{0}1$, et à 0 les autres. L'analyse 2.2(b) des variations de cette fonction sous l'évolution α-asynchrone **FG**(**172**) démontre que, pour toute configuration c :

$$\mathbb{E}[\Delta\Phi] \leqslant -\alpha(|c|_{1100} + |c|_{10100}).$$



(a) Fonction potentiel pour l'analyse de **FG**(**172**): $\Phi(c) = |c|_1 + 2|c|_{101} + |c|_{10100}$.



(b) Certificat du calcul des variations de Φ pour **FG**(**172**).

Analyse 2.2: **FG**(**172**)

Le lemme suivant nous permet alors de conclure.

Lemme 2.36. *Pour toute configuration c instable pour* **FG**(**172**) *et contenant le motif* 00*,*

$$|c|_{1100} + |c|_{10100} > 0$$

Démonstration. Le sous-arbre enraciné en 1̇0̇0 de l'arbre de l'analyse 2.2(a), démontre la

relation : $|c|_{100} = |c|_{1100} + |c|_{10100} + |c|_{00100}$ pour toute configuration c. Procédons par contraposée et supposons que $|c|_{1100} + |c|_{10100} = 0$. Alors, $|c|_{100} = |c|_{00100}$ et toutes les cellules correspondant au masque $\dot{1}00$ correspondent en fait à $00\dot{1}00$. Il s'en suit que si $c \neq 0^n$, comme c contient par hypothèse un motif 00, c contient un motif 100 et en parcourant la configuration vers la gauche depuis ce motif, on en déduit que c est constituée de 1 isolés séparés par au moins deux 0. c est donc stable pour la règle **FG**(**172**). ■

Ainsi, tant que c^t n'est pas stable, $\mathbb{E}[\Delta\Phi_{t+1}] \leqslant -\alpha$. Or, comme $\Phi_t = 0 \Rightarrow c^t = 0^n$, le lemme 2.19 implique que c^t est stable après au plus $O(\frac{n}{\alpha})$ pas en espérance, soit au bout d'un temps normalisé $O(n)$.

Remarquons qu'en espérance, au bout d'un temps normalisé $\Theta(\frac{1}{1-\alpha})$, la configuration initiale 01^{n-1} converge vers la configuration 1^n avec probabilité $\frac{1}{2}$, ou bien évolue vers la configuration 001^{n-2} avec probabilité $\frac{1}{2}$ également. Dans ce second cas, la convergence vers la configuration 10^{n-1} a lieu en espérance au bout d'un temps normalisé $\Omega(n)$. L'espérance du temps de convergence normalisé pour la configuration initiale 01^{n-1} est donc $\Omega(\frac{1}{1-\alpha}+n)$.

Nous pouvons alors conclure le résultat énoncé au théorème 2.34 : le temps de relaxation normalisé de l'automate **FG**(**172**) en dynamique α-asynchrone est exactement $\Theta(\frac{1}{1-\alpha} + n)$.

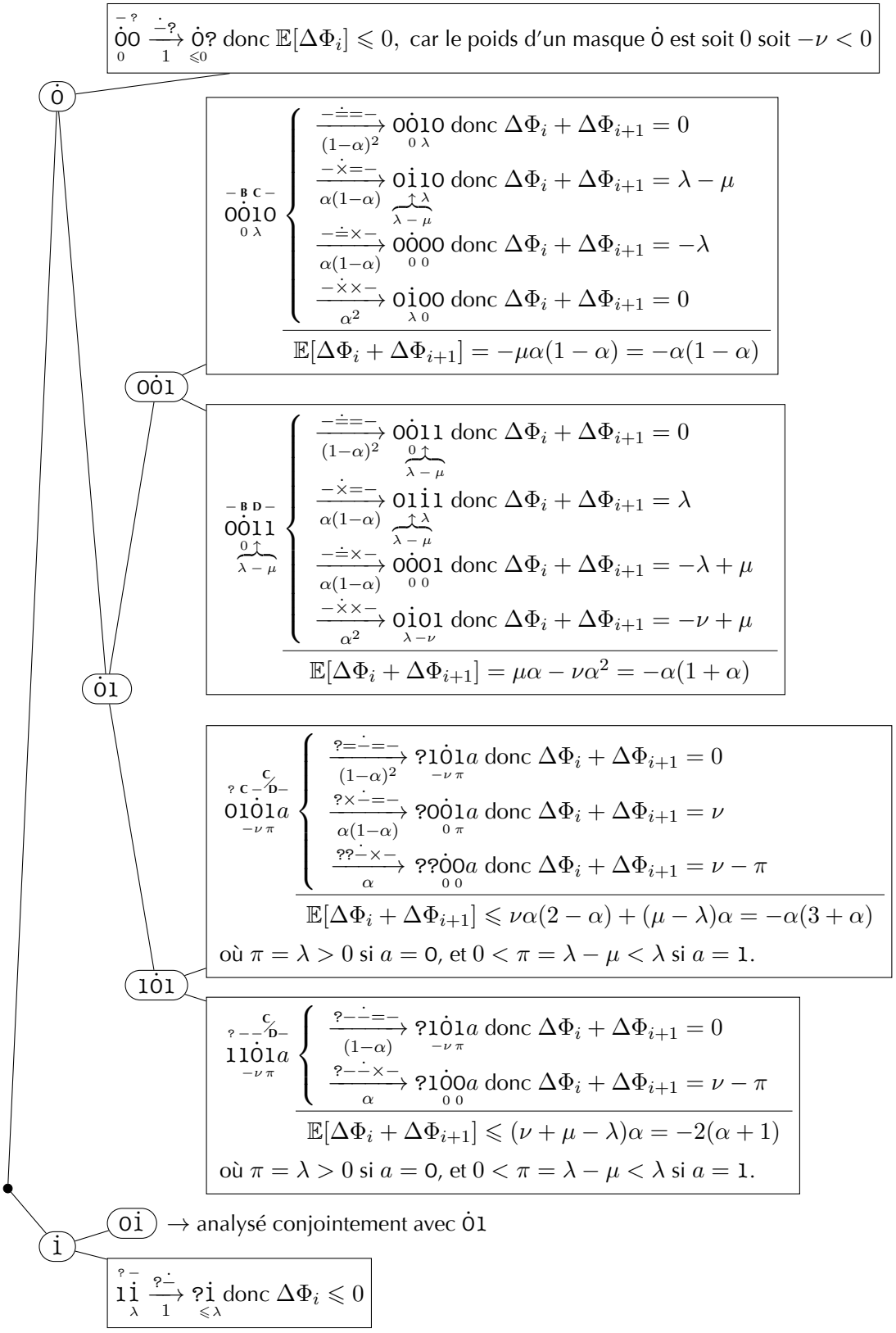
Les marches aléatoires biaisées. La plupart des automates se comportant comme des marches aléatoires biaisées s'analysent de la même façon que les automates monotones. En revanche, le cas de l'automate **BCD**(**194**) est plus intéressant puisqu'il démontre que pour certains automates le régime totalement asynchrone *n'est pas* la limite pour $\alpha \to 0$ du régime α-asynchrone. En effet, le temps de relaxation normalisé de cet automate est $\Theta(n^2)$ en régime totalement asynchrone, et compris entre $\Omega(n)$ et $O(\frac{n}{\alpha(1-\alpha)})$ en régime α-asynchrone. Cette discontinuité sur l'exposant de n en $\alpha = 0$ est due au fait que deux cellules voisines peuvent se mettre à jour simultanément lorsque $\alpha > 0$, chose impossible pour $\alpha = 0$. En régime totalement asynchrone, les frontières 01 suivent des marches aléatoires non-biaisées tandis que les frontières 10 sont immobiles. En régime α-asycnhrone, les mises-à-jour simultanées des voisinages **B** et **D** permettent la fragmentation des frontières 01, fragmentation qui bloque leurs mouvements vers la gauche et biaise leur progression vers la droite, accélérant ainsi la convergence de l'automate de quadratique à linéaire en la taille de la configuration (v. illustration figure 2.10(d)).

Posons $\nu = 1 + \frac{2}{\alpha}$, $\mu = 1$ et $\lambda = 2 + 2\nu$. La fonction potentiel utilisée est décrite par l'arbre de la figure 2.13 page 100. L'intuition est la suivante :

- chaque création d'une nouvelle 1-région doit être sanctionnée, d'où le poids λ pour les masques $\dot{1}0$ et $1\dot{1}1$;
- la frontière gauche de chaque 1-région ayant tendance à aller vers la droite, on donne un poids moindre, $\lambda - \mu$, au masque $0\dot{1}1$;
- enfin, la fragmentation bloquant l'extension vers la gauche des 1-région, on donne un poids très favorable, $-\nu$, au masque $1\dot{0}1$.

L'analyse 2.3 page 108 des variations de Φ sous l'évolution α-asynchrone de **BCD**(**194**) démontre que pour toute configuration c :

$$\begin{aligned}\mathbb{E}[\Delta\Phi(c)] &\leqslant -\alpha(1-\alpha)\big(|c|_{0010} + |c|_{0011} + |c|_{0101} + |c|_{1101}\big)\\ &= -\alpha(1-\alpha)|c|_{01}\end{aligned}$$



Analyse 2.3: **BCD**(**194**)

Il s'en suit que si $c^0 \neq 1^n$, tant qu'il existe des 1 dans c^t, $\Delta\Phi_{t+1} \leqslant -\alpha(1-\alpha)$. Or comme Φ prend ses valeurs dans l'intervalle $[0, 2n(2+\frac{2}{\alpha})]$, et que $\Phi_t = 0 \Leftrightarrow c^t = 0^n$, le lemme 2.19 assure que la configuration atteindra la configuration 0^n au bout de $O(\frac{n}{\alpha^2(1-\alpha)})$ pas en espérance. Le temps de relaxation normalisé de cet automate est donc $O(\frac{n}{\alpha(1-\alpha)})$ (théorème 2.34).

Remarquons que nous ne connaissons pas actuellement la valeur exacte du temps de relaxation de **BCD**(**194**). En effet l'étude d'une configuration initiale contenant une unique 1-région ne simplifie pas l'analyse puisque cette région peut se fragmenter et engendrer un nombre arbitraire de 1-régions. Aussi la seule borne inférieure que nous ayons est $\Omega(n)$, car au moins n/α pas en espérance sont nécessaires pour éliminer la 1-région initiale de la configuration $1^{n-1}0$. Il serait très intéressant de déterminer la dépendance exacte en α de ce temps de relaxation pour statuer définitivement sur la limite $\alpha \to 0$ de cet automate.

Marches aléatoires non-biaisées. Pour cette classe représentée par les automates **BCG**(**138**), **BCFG**(**170**) et **CEFG**(**184**), à l'instar du régime totalement asynchrone, nous construisons à l'aide d'arbres de masques asynchrone des fonctions potentiel similaires à des martingales avec des probabilités positives de bouger, *i.e.* des processus de type I et II de la section 3.3 [112]. Ces fonctions potentiel permettent d'obtenir des majorants du temps de relaxation normalisé en $O(\frac{n^2}{1-\alpha})$. En revanche, nous sommes pour l'instant incapables de produire des minorants satisfaisants pour ces automates. Nous disposons uniquement de minorants triviaux, $\Omega(n)$. La raison est qu'en régime α-asynchrone, le nombre de régions peut croître par fragmentation ou dédoublement. Ceci rend l'analyse de l'évolution de configurations initiales particulières aussi difficile que le cas général et notre formalisme ne semble pas pour l'instant nous permettre d'obtenir des minorants aussi précis que nos majorants. En effet, un majorant de la décroissance locale de la fonction potentiel sur certains motifs suffit pour en déduire sa décroissance globale, car l'existence d'un seul motif décroissant suffit. En revanche, pour obtenir un minorant de la vitesse de la décroissance de la fonction potentiel, il faut non seulement disposer d'un minorant local de sa décroissance sur tous les motifs possibles mais aussi d'un majorant du nombre des motifs où ces décroissances locales ont lieu. Or, obtenir une majoration du nombre de chaque motif requiert de disposer d'une description globale précise de la configuration, chose difficile à réaliser en régime α-asynchrone, car la configuration se complexifie très rapidement avec le temps du fait de la multiplication du nombre de régions. Remarquons que l'étude de configurations particulières dans ces conditions requiert l'utilistation d'outils souvent extrêmement sophistiqués sortant de nos compétences actuelles et qu'il arrive souvent que l'étude de l'évolution d'une unique configuration initiale prennent en mathématique plusieurs années et autant d'articles avant d'aboutir (v. p. ex. le cours de Ferrari de 2008 à l'Institut Henri Poincaré sur les processus d'exclusion [114]).

Automates divergents. Pour ces automates représentés par **BG**(**142**) et **EG**(**156**), les configurations stables ne sont pas accessibles depuis les configurations instables, ce qui les conduit à rester en perpétuelle évolution. Pour certains d'entre eux, l'ensemble limite n'est pas l'ensemble des configurations non-triviales tout entier. Nous vous renvoyons à la thèse de Damien Regnault [254] pour une description précise de leurs ensembles limites.

Le cas de deux automates plus compliqués : BDG(134) et BCDEFG(178). Les résultats que nous allons présenter rapidement dans ce paragraphe ont été obtenus par Damien Regnault durant sa thèse. Ces travaux sont intéressants car l'analyse de **BDG**(**134**) démontre l'utilité de l'approche par arbre de masques lorsque le nombre de cas explose, et celle de **BCDEFG**(**178**) la possible nécessité de recourir à des méthodes radicalement différentes.

L'analyse de la dynamique α-asynchrone de l'automate **BDG**(**134**) (figure 2.10(e)) se révèle

étonnamment difficile. Ses configurations stables sont 0^n, 1^n et $(01)^{n/2}$ (lorsque n est pair). En régimes synchrone ou totalement asynchrone, cet automate diverge depuis toute configuration instable. En effet, en régime totalement asynchrone, le nombre de régions ne peut décroître car les voisinages **C** (010) et **F** (101) sont inactifs. La convergence en régime α-asynchrone est rendue possible par la double-délétion des motifs 0110 (mises-à-jour simultanées des voisinages **D** et **G**). Cependant le phénomène de fragmentation des frontières 0011 (mises-à-jour simultanées des voisinages **B** et **D**) tend à multiplier le nombre de 1-régions. On observe alors un phénomène intéressant qui fait que la combinaison de ces deux phénomènes conduit pour cet automate à une disparition rapide des 1-régions, au moins lorsque α est élevé. Le principe est le suivant : les fragmentations morcellent les 1-régions par la gauche, mais ces nouvelles 1-régions bloquent l'activation du 0 isolé en contact avec la 1-région suivante, qui si elle est de longueur 2, va disparaître par double-délétion avec forte probabilité lorsque α est suffisamment proche de 1. On appelle donc *collision* l'apparition du motif 10110 qui va conduire avec une probabilité proche de 1 vers la double-délétion des deux 1 de droite, lorsque α est suffisamment grand.

En considérant une valeur de α suffisamment élevée ($\alpha \geqslant 0.9999$) mais indépendante de n, on peut négliger les cas où moins de deux cellules actives des motifs considérés ne se mettent pas à jour. Le diagramme de la figure 2.16 décrit alors les transitions possibles entre ces différents motifs en ne considérant que les voies où toutes les cellules actives du motif sont mises à jour (flèches épaisses) ou bien où une seule d'entre elles seulement n'est pas mise à jour (flèches plus fines ayant pour origine la cellule qui n'est pas mise à jour). L'étude du diagramme (figure 2.16) révèle que tous ces motifs vont évoluer en majorité vers l'élimination d'une 1-région par double-délétion via l'apparition d'une *collision*. En effet, les fragmentations engendrent typiquement les deux motifs 001010 et 00100110 du groupe ①. Puis,

- les deux 1-régions de ces motifs évoluent préférentiellement vers les motifs du groupe ②;
- les deux 1-régions oscillent alors entre les motifs 00100010 et 001100110, jusqu'à ce qu'une double-délétion se produise ou bien qu'une de deux cellules actives **D** ne se mette pas à jour;
- dans ce dernier cas, on obtient un des deux motifs du groupe ③ où les deux 1-régions oscillent préférentiellement entre les motifs 00110010 et 00100011, jusqu'à ce qu'une double-délétion se produise ou bien que la cellule active **B** ne se mette pas à jour;
- on engendre alors le motif 0010010 du groupe ④ qui évolue ensuite préférentiellement vers les motifs du groupe ⑤, 00$\boxed{10110}$ ou 001$\boxed{10110}$ qui contiennent enfin un motif de collision entraînant préférentiellement une double-délétion de la seconde 1-région.

Il se peut que durant cette évolution l'une des deux 1-régions en engendre une troisième, mais nous verrons que ceci reste minoritaire lorsque α est suffisamment grand. On observe ce processus sur les simulations dès $\alpha = 0.95$ comme le démontre la figure 2.17.

Damien Regnault utilise le diagramme pour pondérer ces différents motifs en fonction de leur distance aux motifs finaux de collision : un poids de 99 pour les motifs des groupes ① et ②, 91 pour ceux du groupe ③, 74 pour ceux du groupe ④ et 64 pour ceux du groupe ⑤. On remarquera le biais de $-2(1-\alpha)$ accordé au motif 001100110 du groupe ② qui permet de se rapprocher significativement des configurations plus favorables avec probabilité $\Theta(1-\alpha)$

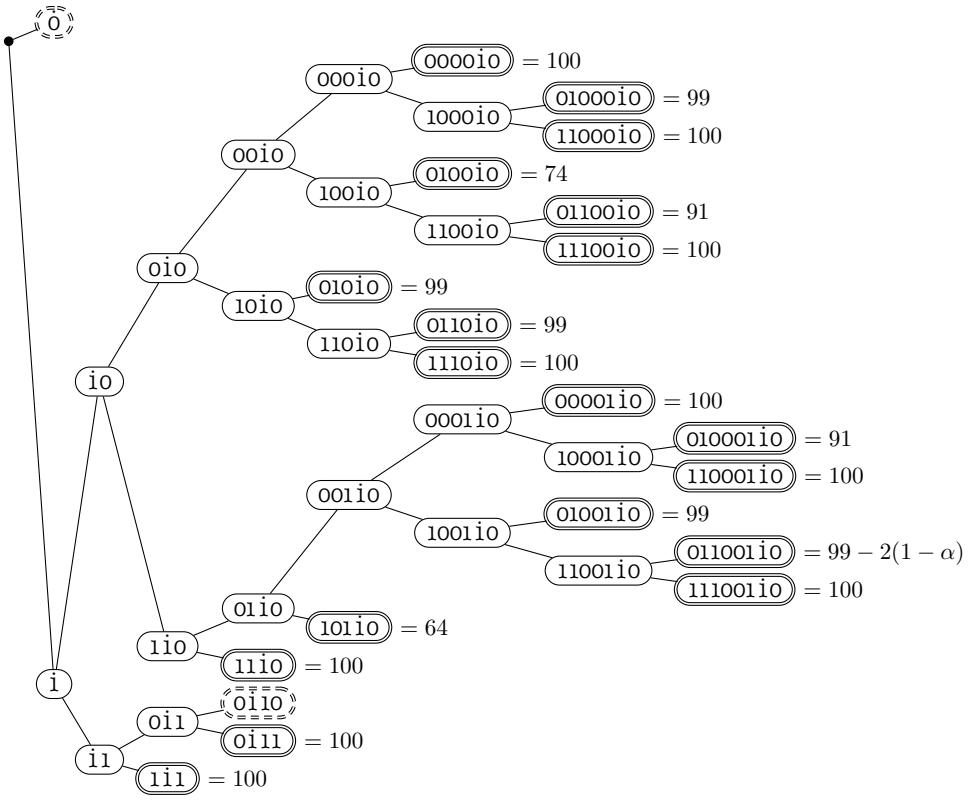


Figure 2.15 – Fonction potentiel pour l'analyse de **BDG**(**134**), extraite de [251, 254].

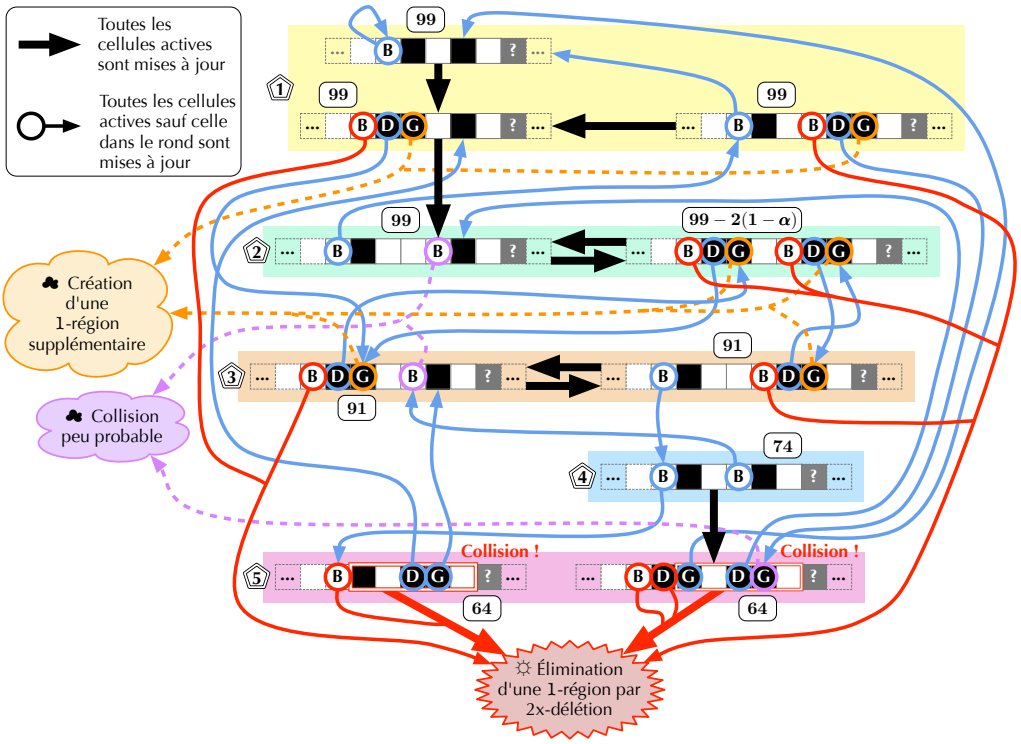


Figure 2.16 – Analyse au premier ordre en $(1-\alpha)$, des transitions entre les motifs menant à des collisions pour la dynamique α-asynchrone de l'automate **BDG**(**134**).

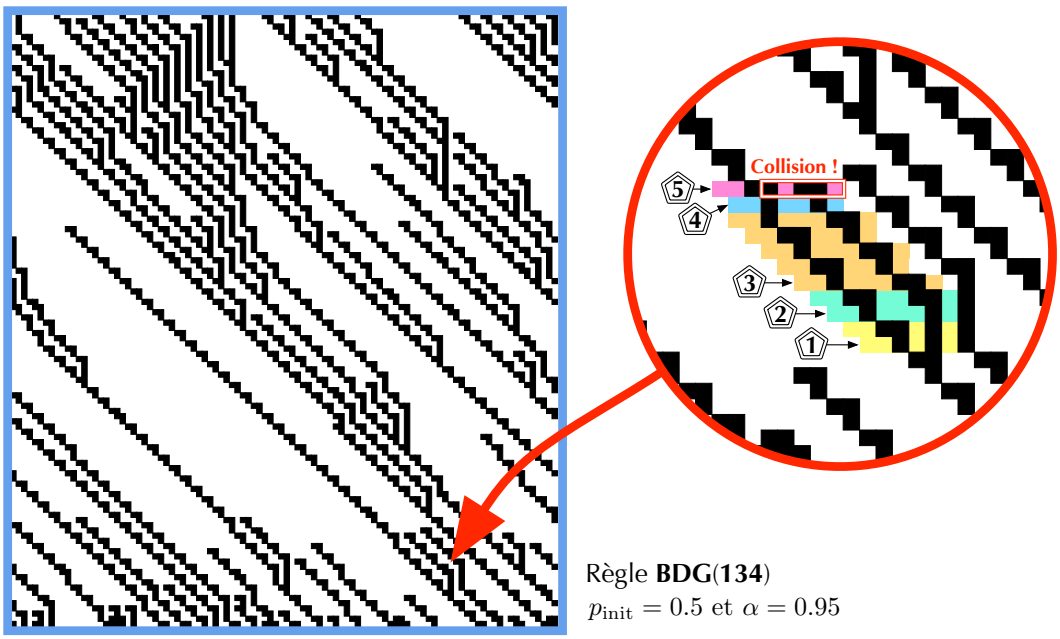


Figure 2.17 – Observation du processus décrit à la figure 2.16 dans une simulation de l'automate **BDG(134)** pour $\alpha = 0.95$.

contrairement à son compagnon du groupe (2). Pour compléter la description de la fonction potentiel, qu'on notera Φ, ces motifs sont ensuite organisés sous la forme d'un arbre de masques (figure 2.15) où le poids des motifs introduits en plus est fixé à 100 (à l'exception des masques 0̇ et de double-délétion 0i̇10, de poids nul). L'(impressionnant) arbre d'analyse 2.4 page 114 permet alors d'énumérer tous les voisinages à étudier en majorant brutalement les cas où plusieurs cellules actives ne se mettent pas à jour. Cette figure est sans doute reproduite ici à une échelle trop petite pour être exploitable. Cependant, nous la donnons uniquement pour démontrer la capacité de ce formalisme à traiter élégamment des situations où les énumérations de cas se révèlent délicates. Aussi, nous n'avons pas jugé sa parfaite lisibilité indispensable au propos de ce manuscrit (le lecteur curieux pourra agrandir à l'envi le fichier PDF pour accéder à tous les détails). En sommant sur toutes les feuilles de l'arbre, Damien Regnault obtient un majorant de l'espérance des variations de Φ, qu'il simplifie à l'aide d'identités obtenues par le corollaire 2.31. Il en déduit ainsi que Φ est bien décroissante en espérance pour α suffisamment grand :

Lemme 2.37 ([251, 254]). *Si* $\alpha \geqslant 0.9999$, $(\forall c)\, \mathbb{E}[\Delta\Phi(c)] \leqslant -0.3(1-\alpha)|c|_{110}$.

Remarquant que toute configuration instable ne contenant que des 1 isolés contient nécessairement un voisinage actif **B** et évolue donc avec probabilité au moins α vers une configuration contenant le motif 11, Damien Regnault conclut dans sa thèse que tant que c^t est instable, $\mathbb{E}[\Phi(c^{t+2}) - \Phi(c^t)] \leqslant -0.3\alpha(1-\alpha)$. Comme $\Phi(c) \in [0, 100n]$ et $(\Phi(c) = 0 \Rightarrow c = 0^n)$, le processus atteint une configuration stable au bout d'au plus $O(\frac{n}{\alpha(1-\alpha)})$ pas de mises-à-jour par le lemme 2.19.

Théorème 2.38 ([251, 254]). *Le temps de relaxation normalisé de* **BDG(134)** *vaut* $O(\frac{n}{1-\alpha})$ *lorsque* $\alpha \geqslant 0.9999$.

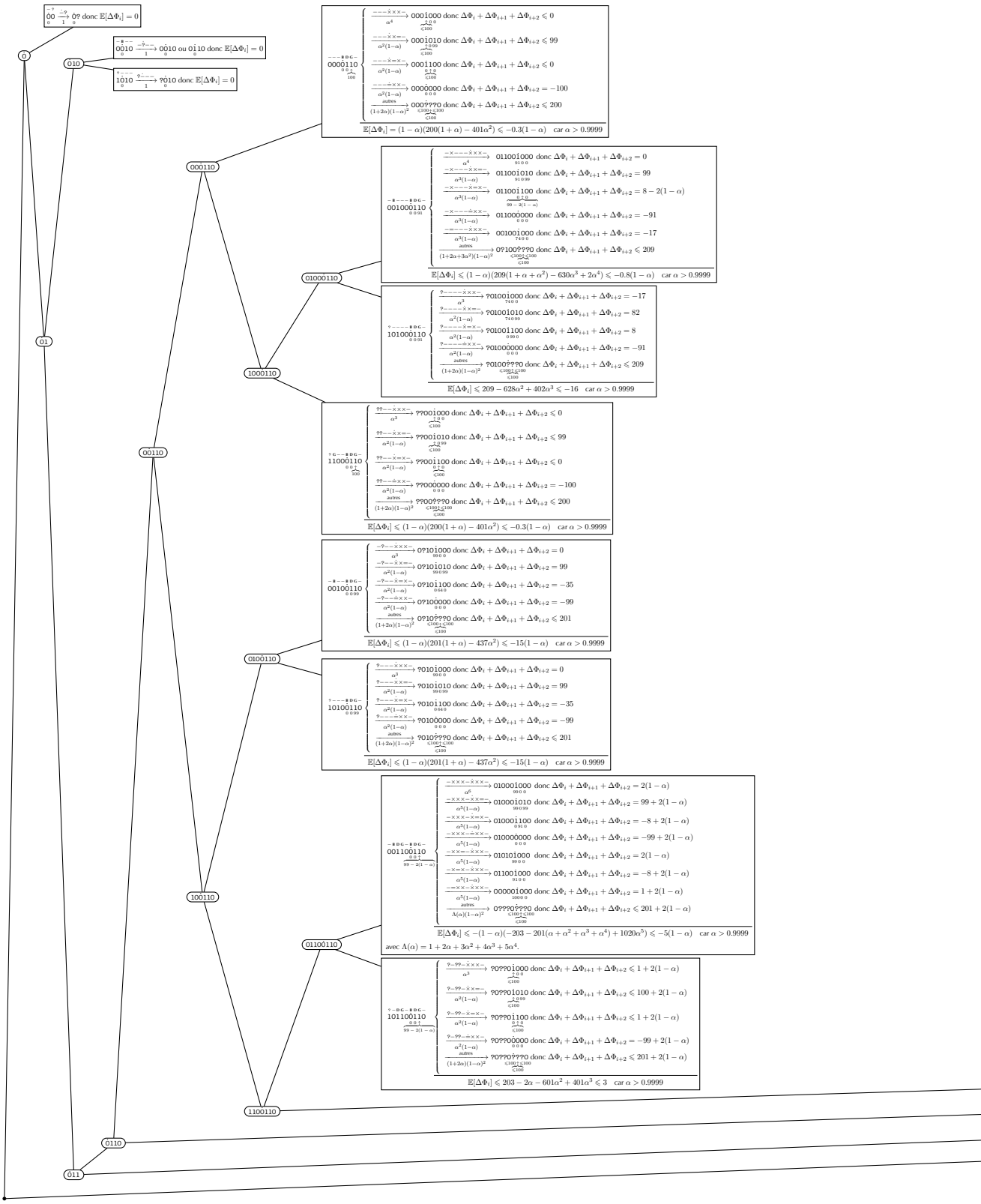
On voit sur cet exemple l'utilité des arbres de masques pour obtenir et analyser des fonctions potentiel dans des situations plus difficiles. On pourra également remarquer la concision du certificat des variations de Φ qui permet de s'assurer très rapidement de la correction du calcul.

Cet exemple démontre également que l'analyse de certains automates nécessite une compréhension très précise des mécanismes en jeux et que ces mécanismes peuvent prendre la forme de chaînes de Markov étonnamment évoluées. En revanche, une fois ce mécanisme compris, notre formalisme offre un cadre sûr pour développer une fonction potentiel s'appuyant sur cette description et en développer l'analyse des variations. Remarquons qu'il est évidemment possible d'améliorer légèrement la borne de 0.9999 en détaillant toutes les évolutions possibles des motifs dans l'arbre d'analyse 2.4 (p. ex. avec l'assistance de `Maple`), sans majorer brutalement l'ensemble des évolutions impliquant plusieurs cellules actives non-mises-à-jour. Cependant, on ne peut espérer qu'une amélioration marginale de cette borne. Comme le souligne Damien Regnault dans sa thèse, le diagramme de la figure 2.16 ne tient pas compte des phénomènes se produisant pour les valeurs nettement inférieures de α. Aussi, toute amélioration significative de cette borne (nous conjecturons au vu des simulations que le temps de relaxation normalisé de cet automate est en fait $\Theta(\frac{n}{\alpha(1-\alpha)})$ pour tout $0 < \alpha < 1$) impliquera de complexifier encore ce diagramme et sans doute d'augmenter l'étendue des voisinages étudiés. Il semble évident que l'assistance informatisée sera alors indispensable, ce dont nous débattrons à la fin de ce chapitre.

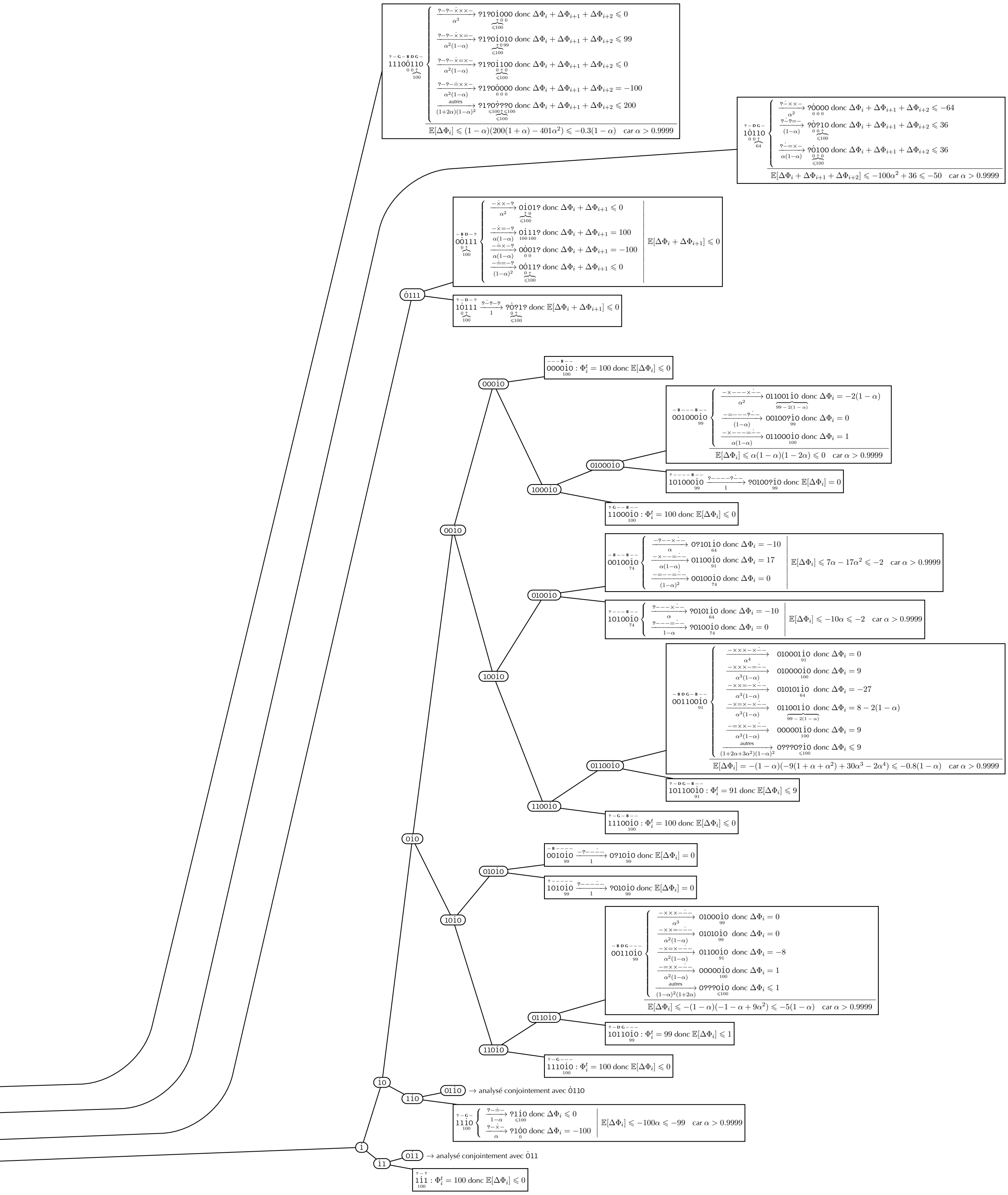
Nous évoquons maintenant rapidement l'étude de l'automate **BCDEFG**(**178**) (v. illustration figure 2.10(i) page 96) effectuée par Damien Regnault durant sa thèse [252, 254]. La règle de transition de cet automate se comprend facilement : une cellule change d'état dès que son voisinage n'est pas homogène, d'où son surnom *flip-if-not-all-equal*. Au vu des simulations, nous conjecturons que le temps de relaxation de cet automate est polynomial (sans doute $\Theta_\alpha(n^2)$) pour $\alpha < 0.5$ et exponentiel pour $\alpha > 0.5$. Nous nous sommes intéressés au régime exponentiel de cet automate. Il est apparu dans nos diverses tentatives pour appréhender le comportement de cet automate que ce processus n'est sans doute *pas dû à des phénomènes locaux*, mais à une dynamique de groupe qui se met progressivement en place dans la configuration. Le formalisme que nous avons développé précédemment, basé sur la pondération de motifs locaux, s'avère alors impuissant pour étudier ce régime.

Il était suspecté depuis longtemps qu'il existe des liens entre certains automates cellulaires probabilistes et les phénomènes de percolations dirigés en mathématique, v. p. ex. la thèse de Nazim Fatès [107]. Aussi, nous avons proposé à Damien de participer à un atelier sur la percolation organisé à Oberwolfach en octobre 2007. Au retour de cet atelier, Damien a pu exhiber un couplage entre le comportement α-asynchrone de cet automate et la percolation dirigée de paramètre p sur le quart de plan.

La *percolation dirigée de paramètre p sur le quart de plan* désigne un processus aléatoire qui ouvre un ensemble aléatoire de sites (v. illustration figure 2.18). Les sites sont les points entiers du quart de plan $y \geqslant \max(x, -x)$ et chaque site possède deux liens qui pointent vers ses deux voisins nord-est et nord-ouest. Chacun de ces liens est *ouvert* indépendamment avec probabilité p. L'*amas initial* désigne l'ensemble des sites accessibles depuis l'origine $(0, 0)$ en ne suivant que des liens ouverts. L'objet de la percolation dirigée est d'étudier la probabilité $\theta(p)$ que l'amas initial soit infini. On démontre (v. [152]) qu'il existe une valeur p_c telle que $\theta(p) = 0$ pour $p < p_c$, et $\theta(p) > 0$ pour $p > p_c$. De plus, on sait que $0.6298 < p_c < 2/3$. Ce processus est illustré à la figure 2.18 où seul l'amas initial est représenté.



Analyse 2.4: Certificat des variations de la fonction potentiel de [251, 254] pour **BDG**(**134**).


0111
010
10
110
1
11
→ analysé conjointement avec 0110
→ analysé conjointement avec 011
$\mathbb{E}[\Delta\Phi_i + \Delta\Phi_{i+1}] \leqslant 0$
$\mathbb{E}[\Delta\Phi_i] \leqslant -100\alpha \leqslant -99$ car $\alpha > 0.9999$
$\mathbb{E}[\Delta\Phi_i + \Delta\Phi_{i+1} + \Delta\Phi_{i+2}] \leqslant -100\alpha^2 + 36 \leqslant -50$ car $\alpha > 0.9999$
$\mathbb{E}[\Delta\Phi_i] \leqslant (1-\alpha)(200(1+\alpha) - 401\alpha^2) \leqslant -0.3(1-\alpha)$ car $\alpha > 0.9999$

Dans [252, 254], Damien Regnault démontre que si $\alpha \geqslant \alpha(p) = \sqrt[3]{1-(1-p)^4}$, il est possible de coupler les mises-à-jour α-asynchrones de l'automate **BCDEFG**(**178**) depuis la configuration bi-infinie $\cdots 00100 \cdots$ et l'ouverture des liens entre les sites de la percolation dirigée de paramètre p, de sorte que si le site (i,t) appartient à l'amas initial, la cellule i est active au temps t dans l'automate. Il s'en suit que pour $\alpha > \alpha(p_c)$, et donc en particulier pour $\alpha > \alpha(2/3) = \sqrt[3]{80/81} \doteq 0.99587$, l'automate **BCDEFG**(**178**) a une probabilité non-nulle de ne jamais atteindre de configuration stable puisque l'amas auquel il est couplé est infini avec probabilité strictement positive (v. figure 2.18).

Théorème 2.39 ([252, 254]). *Si* $\alpha > 0.996$,

$$\Pr\{\textbf{BCDEFG}(\textbf{178}) \textit{ ne converge pas depuis } \cdots 00100 \cdots\} > 0$$

Ainsi lorsque α est élevé, la moindre instabilité se propage pour cet automate dans toute la configuration. Ceci ne démontre pas à proprement parler que le temps de relaxation normalisé de **BCDEFG**(**178**) est exponentiel sur des configurations finies pour $\alpha > 0.99587$. Le résultat de Damien permet en revanche à Irène Marcovici de conclure dans son mémoire de mastère [205] qu'il existe une mesure invariante non-triviale (c.-à-d., non-réduite à une combinaison des deux configurations stables de l'automate, tout-blanc et tout-noir) pour cet automate sur des configurations bi-infinies. L'objet du mémoire de mastère d'Irène Marcovici est d'étudier l'existence d'une transition de phases sur les mesures invariantes de cet automate. Il est en effet conjecturé qu'il existe une valeur $0 < \alpha_c < 1$ telle qu'il n'existe pas de mesure invariante non-triviale pour l'automate **BCDEFG**(**178**) en régime α-asynchrone lorsque $\alpha < \alpha_c$ mais qu'elle existe pour $\alpha > \alpha_c$. Une part importante de son travail a consisté à développer un algorithme de simulation, basé sur la méthode du couplage par le passé (introduite par [243]), pour tester cette conjecture.

Il me semble inévitable de faire un parallèle avec notre conjecture sur l'existence d'une transition de phase polynomiale/exponentielle sur le temps de relaxation de ce même automate sur des configurations finies. Il me semble très raisonnable de penser que ces deux conjectures sont en fait équivalentes et il serait très intéressant de le démontrer, ne serait-ce que parce qu'il est bien plus simple par simulations d'obtenir une excellente estimation du temps de relaxation sur des configurations finies que de statuer sur l'existence ou non d'une mesure invariante non-triviale. Les raisons pour lesquelles je suis convaincu de cette équivalence reposent sur l'intuition suivante : un temps de relaxation polynomial signifie que la configuration tend rapidement à revenir vers une des deux configurations finies stables, il est donc raisonnable de penser que sur une configuration infinie, toute distribution qui différerait d'une combinaison des deux états tout-blanc et tout-noir tendra à s'en rapprocher sur des segments de plus en plus grands et ne pourra donc être stable ; inversement un temps de relaxation exponentiel implique qu'il existe des configurations initiales pour lesquelles des cellules deviennent actives un nombre exponentiel de fois en la longueur du segment considéré, il est donc raisonnable de penser que sur une configuration infinie, ces mêmes cellules redeviendront régulièrement actives, et que la configuration restera confinée dans un espace éloigné des deux configurations stables. Ces suppositions sont bien sûr très vagues et il est possible que cette équivalence ne soit vérifiée que dans des cas très précis seulement, voire même jamais. Cependant, si elle était vraie, cette équivalence créerait un lien fort intéressant entre un paramètre algorithmique naturel, le temps de relaxation, et un objet d'étude traditionnel de la théorie des probabilités, les mesures invariantes, ce qui justifie qu'on s'y intéresse de près. Nous étudions actuellement les moyens de prouver cette équivalence, même partiellement.

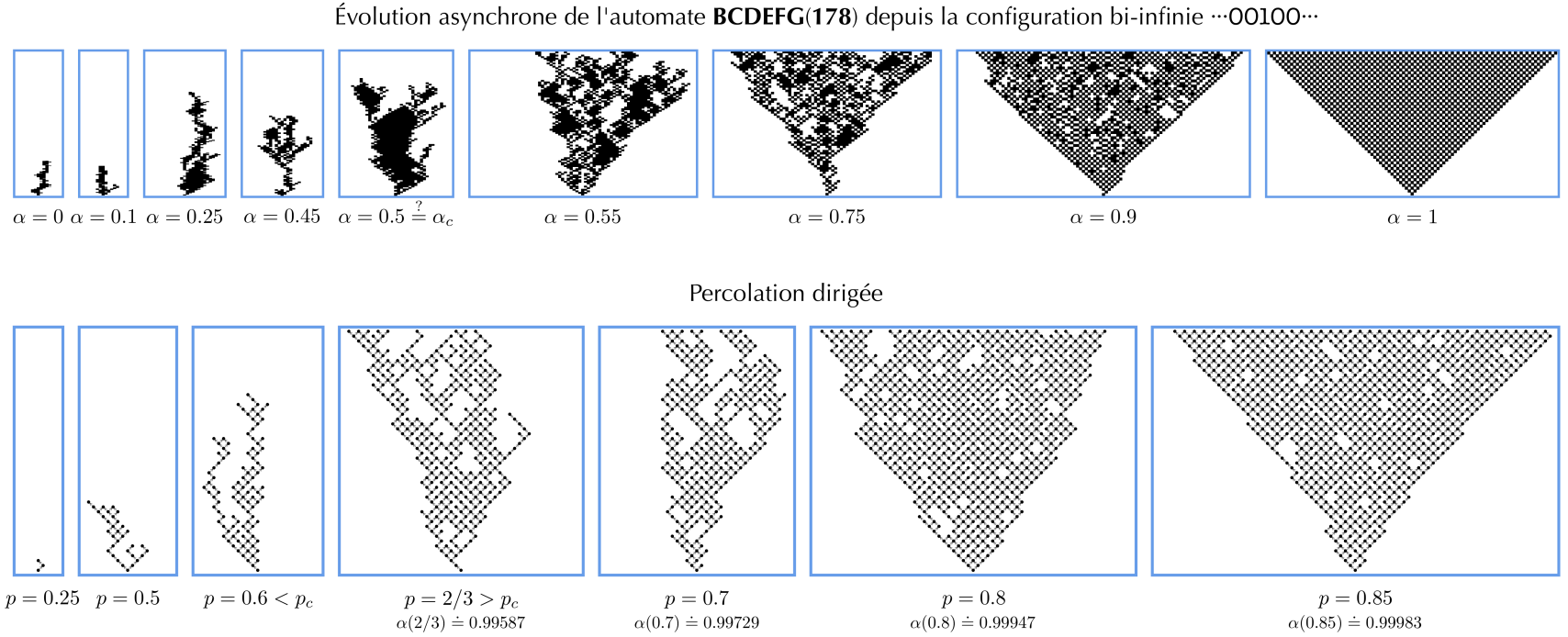


Figure 2.18 – **BCDEFG**(**178**) et percolation dirigée.

Notons que nous réévoquerons l'automate **BCDEFG**(**178**) dans la section suivante qui est consacrée à l'étude de l'automate Minorité 2D. L'automate **BCDEFG**(**178**) est en effet équivalent à l'automate Minorité 1D **ABDEGH**(**23**), au sens que l'on passe de l'un à l'autre, en prenant le *ou exclusif (XOR)* de la configuration avec la configuration alternée $(\texttt{01})^\infty$.

3.6 Conjectures

Contrairement au cas totalement asynchrone, où le temps de relaxation normalisé est connu pour chaque automate, déterminer sa valeur en régime α-asynchrone présente un nombre important de difficultés supplémentaires pour quelques uns de ces automates. Nous vous présentons ici quelques conjectures basées sur nos simulations ainsi que quelques avancées vers leurs démonstrations.

Conjectures 2.40. *Parmi les douze DQECA aux comportements α-asynchrones non-déterminés pour $0 < \alpha < 1$,*

- *trois, représentés par* **BCDEG**(**146**) *et* **BCDEFG**(**178**)*, admettent une transition de phase* $\widehat{\text{TR}}$ *polynomial/exponentiel autour d'une valeur critique α_c du taux d'asynchronisme, avec $\alpha_c^{\mathbf{146}} \approx 0.66$ et $\alpha_c^{\mathbf{178}} \approx 0.50$; nous pensons qu'on doit pouvoir déduire du résultat Damien Regnault [252, 254] portant sur les configurations bi-infinies, que $\alpha_c^{\mathbf{178}} < 0.996$ pour les configurations finies.*
- *huit, représentés par* **DEFG**(**180**) *et* **BDG**(**134**)*, convergent en temps polynomial grâce aux fragmentations ; Damien Regnault [251, 254] a démontré que c'était le cas pour* **BDG**(**134**) *lorsque $\alpha \geqslant 0.9999$;*
- *le dernier,* **BDEG**(**150**)*, converge "par hasard" au bout d'un temps exponentiel sans que n'émerge de structure dans la configuration ; la dépendance en α du temps de relaxation reste à déterminer.*

Enfin, nous conjecturons que le temps de relaxation normalisé des automates de type marches aléatoires non-biaisées (représentés par **BCG**(**138**)*,* **BCFG**(**170**) *et* **CEFG**(**184**)*) est en $\Omega_\alpha(n^2)$, pour $0 < \alpha < 1$.*

Il nous semble que la compréhension du phénomène de fragmentation est la clé pour démontrer ces résultats. Lorsque ce phénomène domine, la configuration se couvre de motifs alternés 01 et la convergence est au mieux exponentielle. Inversement lorsque ce phénomène est faible, le comportement est assimilable aux marches aléatoires des 0- et 1-régions et la convergence a lieu en temps polynomial.

4 Études en dimension 2

Nous avions presenté nos résultats sur les automates 1D à Cris Moore qui nous a encouragés à poursuivre notre étude en explorant la 2D sur laquelle il considère que très peu de choses sont connues. C'est donc avec la bénédiction de Cris Moore, que nous avons décidé d'étudier comment nos analyses pourraient s'étendre au cas des automates cellulaires asynchrones bi-dimensionels. Le passage à la dimension 2 est réputé difficile. Par exemple, Balister, Bollobás et Kozma démontrent dans [25] que la méthode du champ moyen ne permet pas d'approcher correctement les comportements des automates probabilistes bidimensionnels. Il faut donc tenir compte des structures particulières de la configuration. Les études aussi bien mathématique

qu'expérimentale des automates les plus simples, comme le phénomène de percolation dans l'automate majorité, se révèlent extrêmement complexes à mener [164, 149, 10, 214, 55].

Comme nous l'avons vu à la section 2.5.2, le nombre d'automates 2D à deux états est tout simplement prodigieux et il est désespéré d'envisager une étude systématique, même pour les plus petites relations de voisinage. Aussi, nous nous sommes concentrés sur la classe particulière que nous avons évoquée au début de ce chapitre : les totalisants externes dont les transitions ne dépendent que de l'état courant de la cellule et de la somme des états de ses voisins. Le nombre de totalisants externes reste conséquent : $1\,024$ pour le voisinage de von Neumann et $262\,144$ pour le voisinage de Moore (avant symétries). Aussi, nous avons décidé de nous consacrer à l'étude d'un automate particulier, la règle de minorité. Cet automate présente en effet des caractéristiques partagées avec de nombreux autres automates totalisants.

La règle de minorité fait partie des automates à seuil dont le régime synchrone a été très étudié, en particulier dans le cadre des réseaux de neurones. Dans [142], Goles et Martinez démontrent p. ex. que ces automates convergent toujours en régime synchrone vers des cycles de longueur 1 ou 2 au bout d'un temps fini. Pour la règle de minorité en régime synchrone, on observe l'apparition de tâches noires et blanches clignotantes pour les deux relations de voisinage. Le régime asynchrone de cette règle est en revanche encore largement inexploré, ce qui vaut également pour l'ensemble des automates à retour négatif, c.-à-d. dont l'état de chaque cellule tend à changer à moins d'être stabilisé par son voisinage (la règle de minorité, p. ex.). Dans les simulations que nous avons menées, les automates à retour positif (p. ex., la règle de majorité) ne semblent pas présenter de comportements originaux en régime asynchrone. En revanche, la règle de minorité a retenu notre attention pour plusieurs raisons. Tout d'abord, cette règle, pourtant simple à décrire, présente un comportement d'une étonnante richesse : au moins une transition de phase très nette avec des diagrammes espace-temps radicalement différents au-dessus ou en-dessous d'une valeur critique α_c, et sans doute deux nouvelles transitions de phases au vu des simulations complémentaires que j'ai menées pour la rédaction de ce manuscrit. Ensuite, les diagrammes espace-temps de cette règle présentent malgré tout une structure qui permet d'espérer de pouvoir en analyser le comportement (ce n'est pas le cas de tous les automates totalisants externes en régime asynchrone,v. la fin de cette section). Aussi cette règle au comportement complexe mais structuré nous a semblé être un excellent candidat pour tester l'extension de nos méthodes. En fait, nous sommes à présent convaincus que cette règle est une étape clé vers l'analyse d'une famille bien plus large d'automates totalisant présentant des comportements similaires dans les simulations.

4.1 Les automates vN-et M-Minorité 2D

On considère des configurations $n \times m$ toriques et on note $N = nm$ le nombre total de cellules. On notera $\sim$ la relation de voisinage entre deux cellules : les quatre plus proches voisins pour le voisinage de von Neumann (vN- en abrégé), ou les huit plus proches pour le voisinage de Moore (M- en abrégé). Si on note $\Sigma_{ij} = \sum_{(k,\ell):(i,j)\sim(k,\ell)} c_{k\ell}$ la somme des états des voisins d'une cellule (i, j), la fonction de transition de la règle de minorité est :

$$c'_{ij} := \begin{cases} 1 & \text{si } c_{ij} + \Sigma_{ij} \leqslant \begin{cases} 2 & \text{pour vN-Minorité 2D} \\ 4 & \text{pour M-Minorité 2D} \end{cases} \\ 0 & \text{sinon} \end{cases}$$

Les cellules mises-à-jour ont donc tendance à "choisir" un état en opposition à ceux des cellules voisines.

Nous avons constaté dans nos simulations de divers automates totalisant externes en régime asynchrone que des motifs particuliers émergent dans la configuration et entrent en compétition pour la recouvrir. Ce phénomène est particulièrement présent pour les automates vN- et M-Minorité 2D où des motifs à damier (pour vN-) et des rayures (pour M-) apparaissent pour des taux d'asynchronisme faibles alors que des motifs unis mais bruités apparaissent pour ces deux automates lorsque le taux d'asynchronisme est élevé (v. illustrations fig. 2.23 et 2.24 page 128). Ce schéma est également présent, avec des variantes, dans de nombreux autres automates totalisants.

Les deux sections suivantes vont nous permettre de définir des paramètres qui nous seront utiles par la suite pour comprendre et analyser l'émergence de ces motifs.

4.1.1 Énergie d'une configuration

Il a été observé à de nombreuses reprises [58, 45, 286] que les mises-à-jour asynchrones des cellules tendent à augmenter la stabilité d'une configuration. Une fonctionnelle naturelle pour mesurer la distance d'une configuration à un état stable pour la règle de minorité est la somme sur toutes les cellules du nombre de ses voisines dans le même état qu'elle. Par analogie avec les modèles d'Ising de ferro- et d'antiferromagnétisme [214], nous appelons cette fonctionnelle, l'*énergie* de la configuration. Notons que des fonctionnelles similaires ont été introduites pour démontrer la convergence de certains automates à retour positif [165, 142].

Définition 2.41 (Potentiel et énergie). *Le* potentiel v_{ij} *de la cellule* (i,j) *est le nombre de ses voisines qui sont dans le même état qu'elle, moins un certain potentiel standard* π_0 *:*

$$v_{ij} = \#\{(k,\ell) : (k,\ell) \sim (i,j) \text{ et } c_{k\ell} = c_{ij}\} - \pi_0$$

Par convention, les potentiels standards sont $\pi_0^{vN} = 0$ *et* $\pi_0^{M} = 2$ *pour vN- et M-Minorité 2D respectivement. L'*énergie *d'une configuration* c *est la somme des potentiels :*

$$E(c) = \sum_{i,j} v_{ij}.$$

Remarquons qu'avec cette convention, l'espérance de l'énergie d'une configuration aléatoire est $2N$ pour les deux relations de voisinage.

Fait 2.42. *Une cellule* (i,j) *est active pour vN- ou M-Minorité 2D si et seulement* $v_{ij} \geqslant 2$. *Quand une cellule active est mise à jour, son nouveau potentiel est* $v_{ij} := 4 - v_{ij}$ *pour les deux relations de voisinage.*

Notons que la mise-à-jour d'une cellule (active) de potentiel 2 laisse son potentiel invariant, nous disons que cette mise à jour est *réversible*. En revanche, la mise-à-jour d'une cellule de potentiel $\geqslant 3$ est *irréversible* car la cellule est inactive après la mise-à-jour (son potentiel devient $\leqslant 1$). Chaque cellule active sera marquée dans les figures d'un rond rouge, si sa mise-à-jour est réversible, ou violet si elle est irréversible.

La proposition suivante justifie l'utilisation du potentiel standard $\pi_0^{M} = 2$, qui assure que les configurations d'énergie minimale sont d'énergie nulle.

Proposition 2.43 (Positivité). *Pour toute configuration* c*,*

$$0 \leqslant E^{vN}(c) \leqslant 4N \quad \text{et} \quad 0 \leqslant E^{M}(c) \leqslant 6N.$$

Le preuve que cette propriété pour M-Minorité 2D utilise un argument de réorganisation locale du potentiel (v. [256, 257]). On notera donc que le potentiel des cellules d'une configuration stable pour M-Minorité 2D n'est donc jamais minimum, ce qu'on appelle le phénomène

de *frustration* en physique. Ce phénomène est à l'origine de différences significatives entre vN- et M-Minorité 2D.

Proposition 2.44 (Minima et Maxima). *Les configurations d'énergie minimale sont :*

- *pour vN-Minorité 2D : les deux damiers pair et impair ;*
- *pour M-Minorité 2D : les quatre configurations rayées horizontalement ou verticalement, et paire ou impaire.*

Les configurations d'énergie maximales sont tout-blanc et tout-noir pour les deux automates.

4.1.2 Structurer les configurations

Afin de comprendre et visualiser l'évolution des automates vN- et M-Minorité 2D, nous allons introduire les définition suivantes qui partitionnent les configurations en régions stables.

Définition 2.45 (Damiers et rayures). *On appelle* damier pair *(resp.,* impair*) la configuration c telle que $c_{ij} = (i+j) \bmod 2$ (resp., $(i+j+1) \bmod 2$) pour $0 \leqslant i < n$ et $0 \leqslant j < m$.*

On appelle rayures horizontales paires *(resp.,* horizontales impaires*,* verticales paires *et* verticales impaires*) la configuration c telle que $c_{ij} = i \bmod 2$ (resp., $(i+1), j, (j+1) \bmod 2$).*

Définition 2.46 (Bords et diamants). *Pour vN-Minorité 2D, on appelle* bord *l'arête commune à deux cellules voisines dans le même état. On représentera les bords d'une configuration de vN-Minorité 2D en* rouge *dans les figures.*

Pour M-Minorité 2D, on dira qu'une cellule (i, j), avec $0 \leqslant i < n$ et $0 \leqslant j < m$, est paire *si $i + j$ est pair, et* impaire *sinon. On dira qu'il y a un* bord *entre deux cellules en diagonales (i, j) et $(i \pm 1, j \pm 1)$ si elles ont le même état, c.-à-d. si $c_{ij} = c_{i\pm1,j\pm1}$. Ce bord est de couleur* bleue *si ces deux cellules (de même parité) sont paires, et* vert *sinon. Le bord est représenté par un segment de longueur $\sqrt{2}$ centré perpendiculairement sur le trait reliant les centres des deux cellules (v. p. ex. les illustrations de la figure 2.20 page 126). On place un* diamant *sur les cellules (i, j) dont l'état ne coïncide pas avec celui des rayures verticales paires, c.-à-d. telles que $c_{ij} \neq j \bmod 2$. Le diamant est* bleu *si la cellule est paire, et* vert *sinon.*

Remarquons que les bords sont une représentation duale de la configuration. Ils la déterminent complètement à symétries près (noir/blanc pour vN-Minorité 2D ; noir/blanc et rotation des rayures pour M-Minorité 2D). Les diamants d'une configuration la déterminent complètement.

Proposition 2.47 (Partition de la configuration [255, 258, 256, 257]). *Pour vN-Minorité 2D, les bords sont exactement les frontières des régions couvertes par les deux damiers pair et impair.*

Pour M-Minorité 2D, les bords sont exactement les frontières des régions couvertes par les quatre types de rayures paires/impaires, et horizontales/verticales. Et si n et m sont pairs, les bords bleus (resp., verts) sont exactement les frontières des régions couvertes par les diamants bleus (resp., verts).[32]

Comme les damiers et les rayures sont des motifs stables de vN- et M-Minorité 2D respectivement, les cellules actives de ces automates se trouvent sur les bords.

[32] L'énoncé de cette proposition est volontairement flou, nous renvoyons le lecteur aux articles [258, 257] pour les énoncés précis.

4.1.3 Configurations stables

Comme les cellules inactives ont un potentiel $\leqslant 1$ pour les deux automates, l'énergie des configurations stables est inférieure à N.

Proposition 2.48 (Énergie des configurations stables). *L'énergie des configurations stables est comprise entre* 0 *et* N.

Au dessus du niveau d'énergie N, aucune configuration n'est stable. En revanche, des configurations stables et instables cohabitent dans les niveaux d'énergie compris entre 0 et N. On parle alors traditionnellement en physique de *phase vitreuse*. Dans une phase vitreuse, les études statistiques locales de la configuration ne donnent aucune indication sur sa stabilité, et l'analyse de l'évolution de ces configurations requiert typiquement de comprendre la structure *globale* de la configuration. La figure 2.19 présentent des configurations de différents niveaux d'énergie typiquement observées pour les deux automates.

Pour vN-Minorité 2D, le potentiel d'une cellule est égal au nombre de bords qu'elle touche. Une cellule est donc inactive si et seulement si elle touche au plus un bord. Ainsi,

Théorème 2.49 (vN-Minorité 2D [255, 258]). *Les configurations stables de vN-Minorité 2D sont les configurations recouvertes par des bandes de damiers de largeur au moins* 2*, toutes verticales ou toutes horizontales. Si* n *et* m *sont impairs, il n'existe pas de configurations stables.*

Les configurations stable d'énergie maximale sont celles pavées par les motifs  *ou*  .

La caractérisation des configurations stables pour le voisinage de Moore est étonnamment plus compliquée et requiert une analyse fine des configurations, similaire aux déductions que l'on peut faire dans le célèbre jeu du *démineur* [318].

Théorème 2.50 (M-Minorité 2D [256, 257]). *Lorsque* n *ou* m *sont pairs, les configurations stables de M-Minorité 2D sont de l'un des trois types suivants (v. fig. 2.20 page 126) :*

- ***Type I :*** *les bords forment des lignes droites diagonales telles que : deux bords de la même couleur sont à distance-*$\ell_1 \geqslant 4$*; et le nombre de bords de chaque couleur coupant chaque ligne (resp., colonne) a la même parité que* m *(resp.,* n*).*
- ***Type II :*** *chaque bord est entrelacé avec un bord de la couleur opposée suivant le motif*  *ou* *; et chaque paire de bords entrelacés est à distance-*$\ell_1 \geqslant 2$ *des autres ; et le nombre de paires de bords entrelacés a la même parité que* n *s'ils sont entrelacés horizontalement, et que* m *sinon.*
- ***Type III :*** *les bords définissent une grille torique bicoloriée rayures horizontales/rayures verticales comme indiqué sur la figure 2.20(a) dont les bords se croisent suivant l'un des motifs de la figure 2.20(d). De plus, aucune configuration stable ne contient à la fois des croisements de type* C *et* D*, et si une région a un croisement de type* C *(resp.,* D*), tous les croisements à la même verticale (resp., horizontale) sont du même type. Enfin, le nombre de colonnes (resp., lignes) de croisements de type* C *(resp.,* D*) a la même parité que* m *(resp.,* n*).*

Réciproquement, toutes les configurations vérifiant ces propriétés sont stables pour M-Minorité 2D. Cet automate n'admet donc aucune configuration stable si n *et* m *sont impairs. De plus, ses configurations stables d'énergie maximale* $E = N$ *sont celles pavées à*

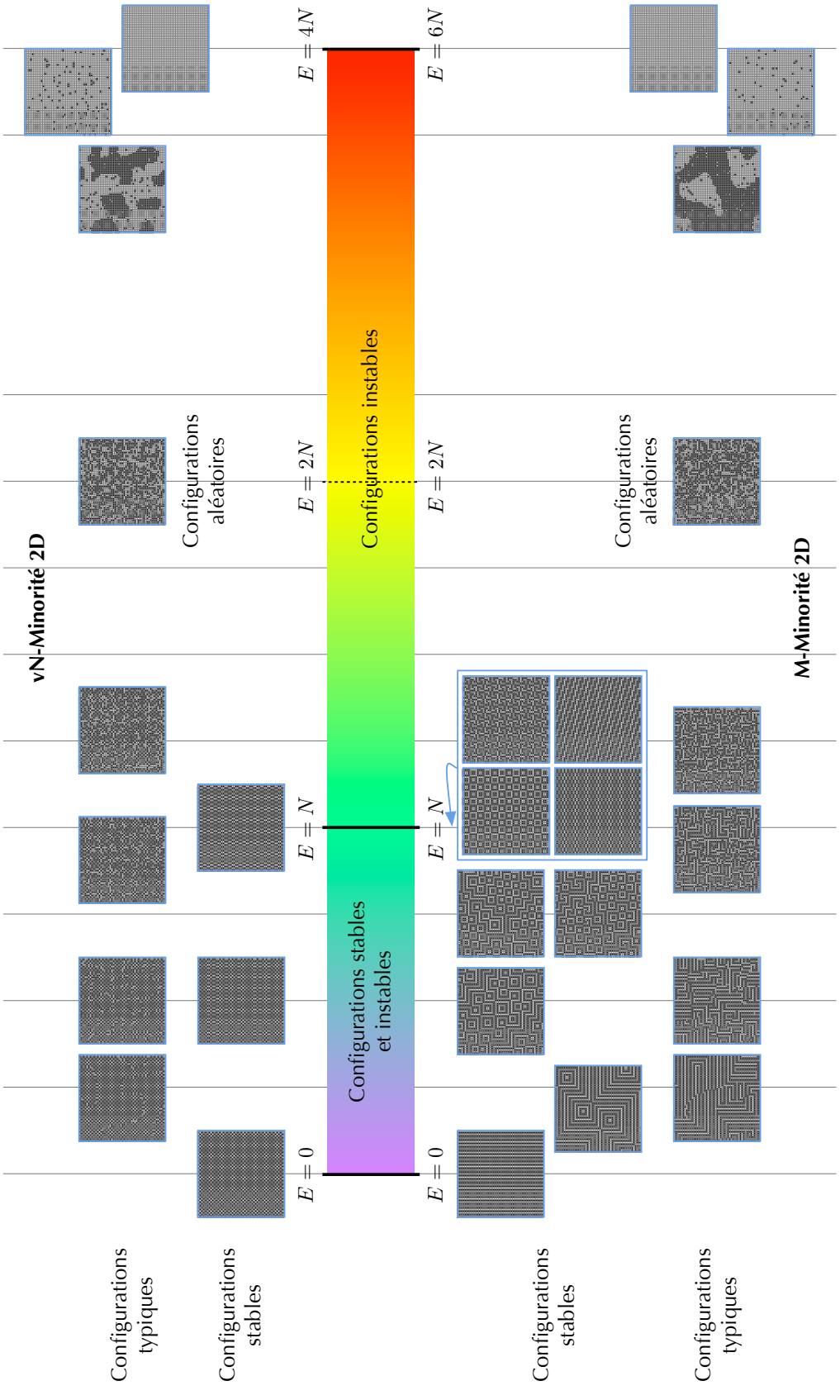
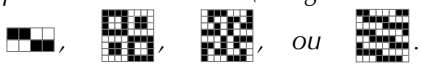


Figure 2.19 – Configurations stables et instables de différents niveaux d'énergie pour les automates vN- et M-Minorité 2D.

symétries près par l'un des quatre motifs suivants (v. Figure 2.19 page 123) :



La figure 2.20 donne des exemples des configurations stables pour M-Minorité 2D. La preuve de ce théorème repose sur de soigneuses chaînes de déductions locales basées sur le fait que le potentiel de chaque cellule ne doit pas dépasser 1 (pour les configurations stables) ou doit égaler 1 (pour celles d'énergie maximale). Se pose alors le problème de l'*écriture* de ces preuves. Il se trouve que ces preuves reposent sur le parcours d'une arborescence couvrante des motifs locaux possibles. Il est alors possible de représenter cette chaîne de déductions en donnant l'ordre de parcours de cette arborescence. On obtient alors la preuve sous la forme de certificats graphiques donnés aux figures 2.21 (pour l'énumération des voisinages compatibles avec les configurations stables) et 2.22 (pour la preuve des configurations stables d'énergie maximale de type III). Nous renvoyons le lecteur à [257] pour le détail de ces preuves.

4.2 Régime α-asynchrone des automates Minorité 2D

Cette section présente nos analyses expérimentales par simulation des comportements asynchrones des automates vN- et M-Minorité 2D. La section suivante sera consacrée à l'analyse mathématique de ces observations. La rédaction de ce manuscrit a été l'occasion de mener une étude expérimentale plus approfondie de ces automates qui a révélé quelques nouveautés intéressantes.

Problèmes posés par la simulation d'automates cellulaires bidimensionnels. Un des freins principaux pour l'étude des automates cellulaires de dimension $\geqslant 2$ est la lenteur de simulations. D'une part, cette lenteur nous contraint à n'explorer que des configurations de tailles raisonnables et sur une infime partie de l'espace des possibles. Il convient donc de se garder de généraliser hâtivement les résultats observés sur les tailles accessibles, d'autant que les évolutions peuvent dépendre très lentement de la taille des configurations (v. la mésaventure de la prédiction du seuil de percolation de majorité par amorce, relatée dans [164, 149]). Le nombre important de cellules mises-à-jour à chaque pas (n^2 pour une configuration $n \times n$) a également la fâcheuse tendance d'épuiser rapidement les ressources des générateurs pseudo-aléatoires qui finissent par boucler avant la fin des simulations. Mais surtout, l'obstacle principal est que la lenteur des simulations réduit considérablement l'interactivité. Il est en effet impossible de stocker l'intégralité des trajectoires calculées (et même si c'était possible, il serait impossible de les visualiser toutes). L'observateur n'accède donc qu'aux valeurs des fonctionnelles qu'il a choisi d'enregistrer durant le calcul des trajectoires. Le moindre changement sur la fonctionnelle observée implique alors de relancer l'intégralité du calcul qui peut s'étaler plusieurs jours. Il faut donc procéder par essais successifs sur de petits nombres de *runs* pour trouver les bonnes valeurs des paramètres à tester et sur quelles durées avant de lancer les simulations à plus grandes échelles.

Notre démarche. Après quelques tâtonnements, nous avons fait le choix de nous restreindre à des simulations sur des configurations de petites tailles, 20×20. Ce choix nous a permis d'étudier avec une interactivité raisonnable (chaque campagne de simulation a pris une dizaine d'heures sur une machine actuelle) les comportements sur un nombre important de trajectoires (un millier par valeur du paramètre étudié), garantissant ainsi une meilleure fiabilité des résultats obtenus et aussi une exploration plus exhaustive de l'espace des possibles. Notons que les

temps de calculs étant en $\Theta(n^4)$ mises-à-jour, passer à des configurations 50×50 ou même 30×30 tue toute possibilité d'interactivité et réduit également paradoxalement la fiabilité des calculs. Nous discuterons à la fin de cette section des moyens de nous assurer que les comportements observés sont bien valables pour des valeurs plus grandes de n.

Dans nos simulations, nous calculons 1 000 trajectoires issues de 1 000 configurations aléatoires 20×20 pour chaque valeur de α testée. Chaque trajectoire est simulée pendant un temps normalisé $\hat{t}_{\max} = 500$.[33] Expérimentalement, les configurations 20×20 qui sont encore instables à cette date, continueront d'évoluer pendant un temps très long que nous présumons exponentiel. Elles présentent en effet une structure très stable dans le temps, et restent dans cet état métastable jusqu'à ce que des mises-à-jour simultanées improbables recouvrent l'ensemble de la configuration par un motif stable, ce qui n'arrive qu'au bout d'un temps exponentiel en espérance (observable uniquement sur des configurations de dimension $n \leqslant 9$).

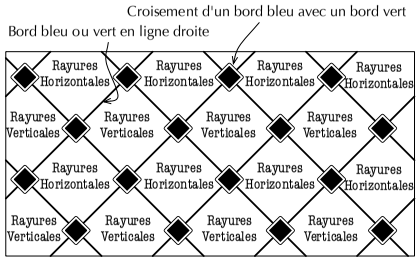
Évolution de l'énergie en fonction du temps normalisé et de α (fig. 2.23 et 2.24). Ces deux figures représentent l'énergie moyenne en fonction de α à différentes dates normalisées allant de $\hat{t} = 0$ à $\hat{t}_{\max} = 500$. On observe en fonction de α deux phases bien distinctes macroscopiquement pour les deux automates vN- et M-Minorité 2D. Pour ces deux automates, nos simulations indiquent l'existence d'une valeur critique α_c, $\alpha_c^{\text{vN}} \approx 0.825$ et $\alpha_c^{\text{M}} \approx 0.57$.

- *Pour $\alpha < \alpha_c$,* l'énergie moyenne de la configuration chute brutalement initialement puis décroît plus lentement pour atteindre une valeur presque nulle. Dans le même temps, les configurations se couvrent très rapidement de régions de motifs stables (damiers pour vN-Minorité 2D, et rayures pour M-Minorité 2D), qui entrent en compétition pour recouvrir la configuration, s'absorbent les unes les autres, jusqu'à ce qu'une configuration stable soit atteinte, au bout d'un temps relativement court ($\lesssim nm$ temps normalisé expérimentalement).
- *Pour $\alpha > \alpha_c$,* l'énergie moyenne croît très rapidement pour se maintenir à des valeurs très élevées pour un temps très long. En effet, les motifs de damier (pour vN-Minorité 2D) et de rayures (M-Minorité 2D) ne semblent plus stables au dessus de α_c et disparaissent très vite. Les configurations se couvrent alors très rapidement de grosses taches unies qui clignotent, mais qui sont progressivement grignotées par un bruit de fond dû aux cellules qui ne se mettent pas à jour, jusqu'à ce qu'il ne reste plus qu'un fond uni clignotant bruité ; cet état métastable semble alors perdurer un temps exponentiel jusqu'à ce qu'une mise-à-jour improbable conduise à un point fixe de l'automate en un pas.

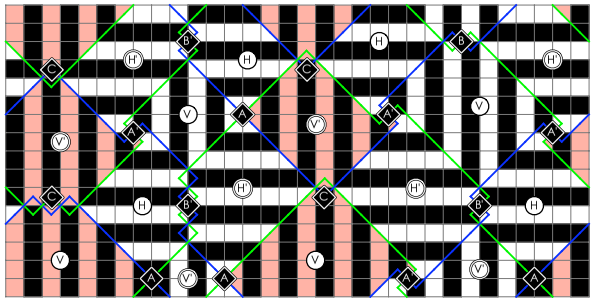
On distingue donc clairement deux régimes bien distincts : une convergence polynomiale vers une configuration stable d'énergie faible, $O(n+m)$, pour $\alpha < \alpha_c$ et une convergence polynomiale vers un état métastable de haute énergie, $\Omega(nm)$, qui perdure pendant un temps exponentiel pour $\alpha > \alpha_c$.

En observant les configurations typiques données en dessous des courbes d'énergie des figures 2.23 et 2.24, on constate que pour $\alpha < \alpha_c$, les configurations sont très structurées pour les deux automates : les bords, peu nombreux, découpent nettement la configuration en larges régions stables. Tandis que pour $\alpha > \alpha_c$, la plupart des cellules des configurations sont instables et les bords ne découpent plus la configuration de façon pertinente. Ces premières expérimentations suggèrent donc qu'il existe une unique transition de phases, entre une phase d'énergie faible, asymptotiquement $O(n+m)$, et une autre d'énergie élevée, $\Omega(nm)$.

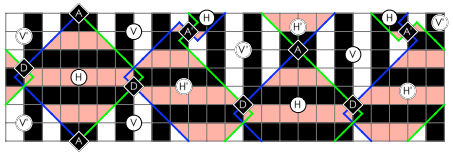
[33] Correspondant à $t_{\max} = \lceil 500/\alpha \rceil$ pas de mises-à-jour si $\alpha > 0$, et $t_{\max} = 500N$ mises-à-jour pour $\alpha = 0$.



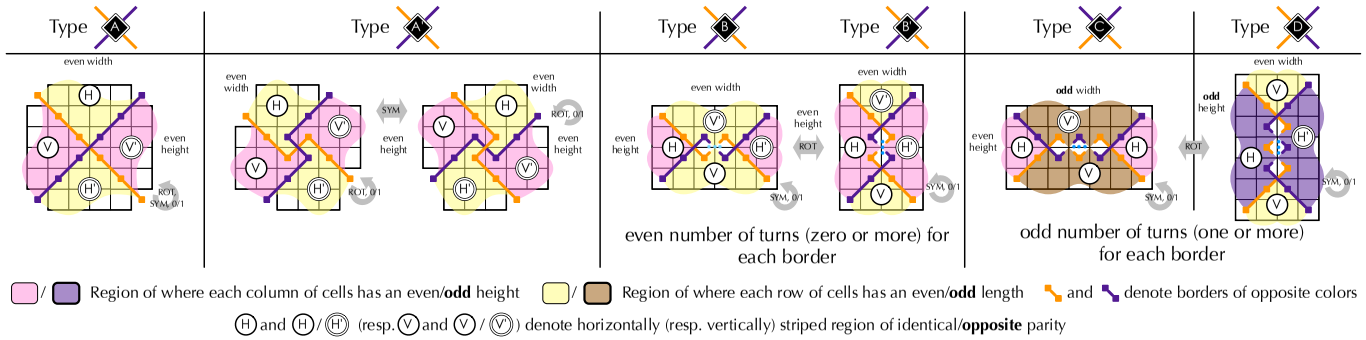
(a) Grille sous-jacente des configurations stables de type III



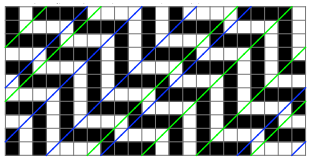
(b) Une configuration stable 16×32 de type III



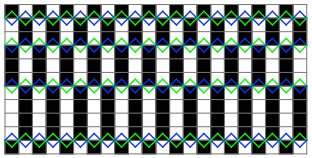
(c) Une configuration stable 7×24 de type III



(d) Liste des croisements autorisés dans les configurations stables de type III



(e) Une configuration stable 11×22 de type I



(f) Une configuration stable 11×22 de type II

Figure 2.20 – Structure des configurations stables de M-Minorité 2D (théorème 2.50)

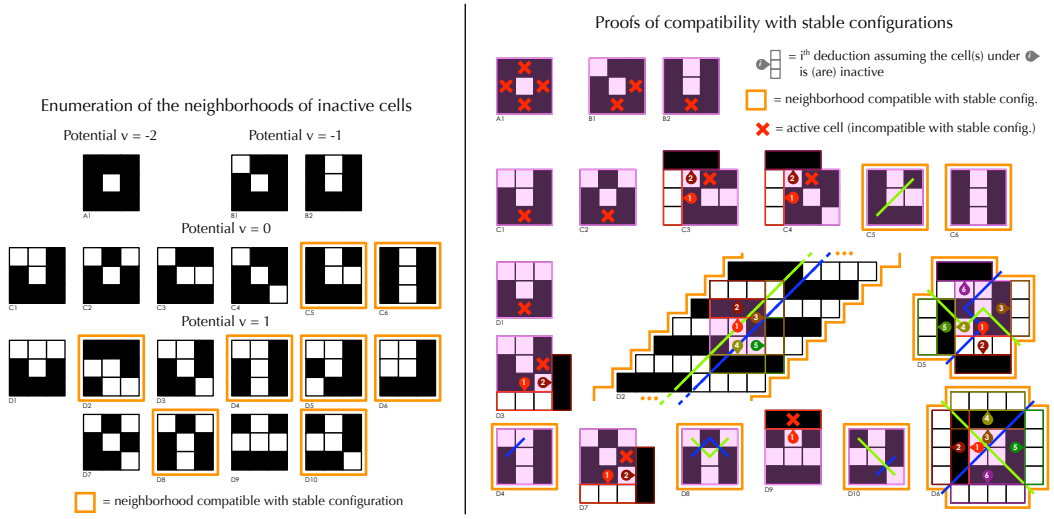
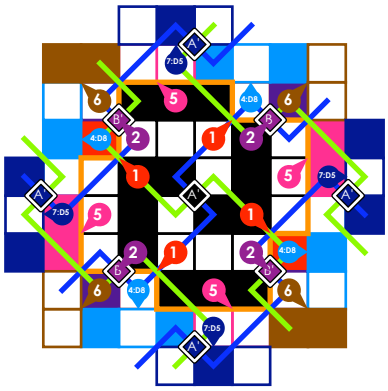


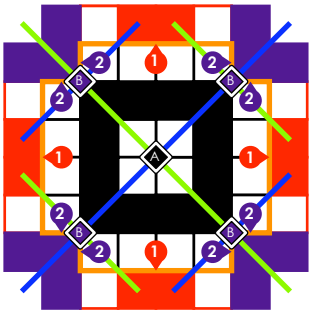
Figure 2.21 – Certificat de l'énumération des voisinages compatibles avec les configurations stables (à symétries près). (preuve de la première partie du théorème 2.50)

the *i*-th deduction based on the fact that the cell(s) under the circle has exactly 3 neighboring cells of its own color

the *i*-th deduction based on the fact that pattern D*k* is the only stable pattern matching the neighborhood of the cell(s) under the circle



(a) Chaîne des déductions pour les configurations ayant un croisement A'.



(b) Chaîne des déductions pour les configurations sans croisement A'.

Figure 2.22 – Chaînes des déductions pour déterminer les configurations de type III d'énergie maximum (preuve de la dernière partie du théorème 2.50).

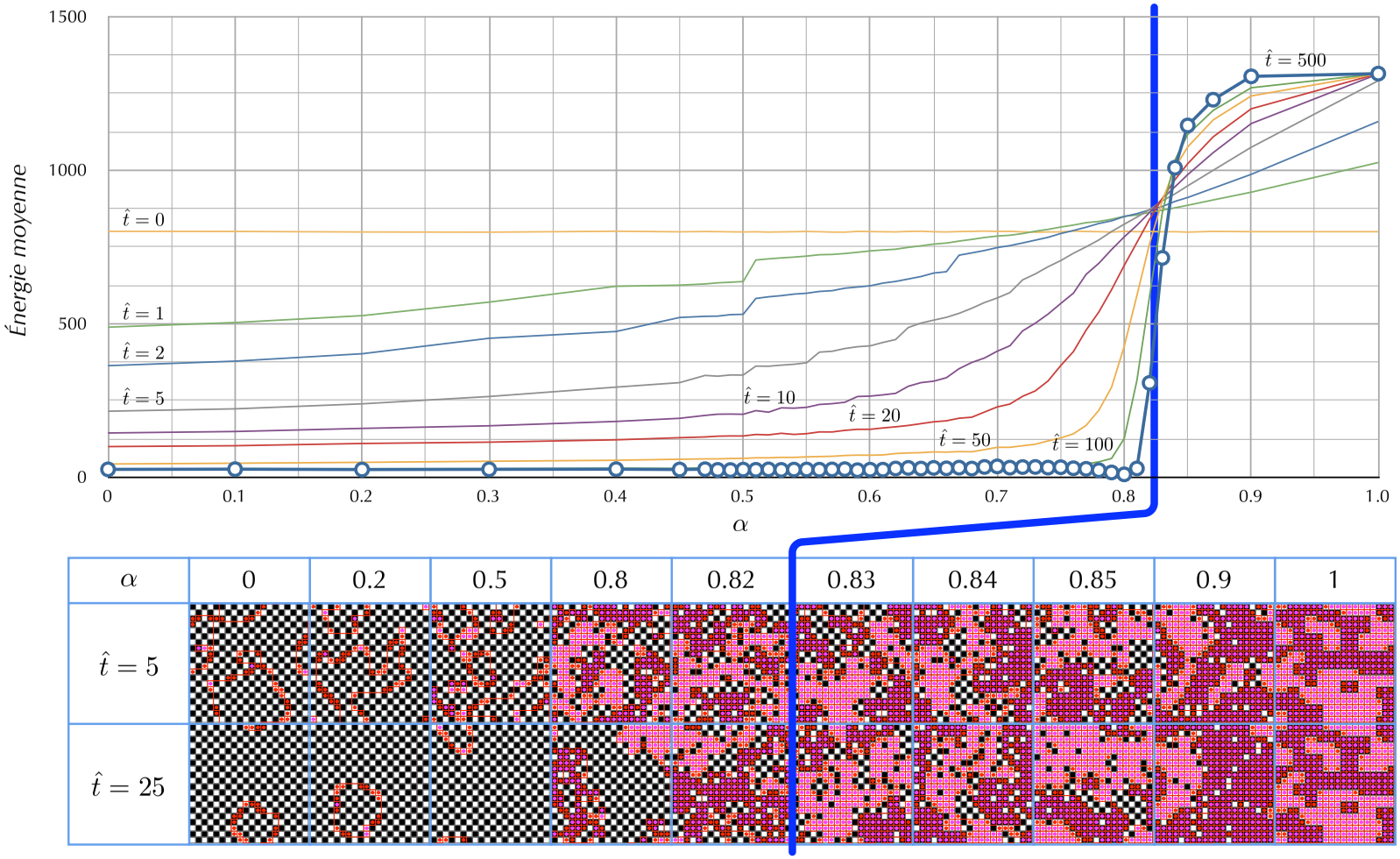


Figure 2.23 – Évolution de l'énergie moyenne en fonction de $\hat{t}$ et de α depuis 1 000 configurations aléatoires 20×20 pour vN-Minorité 2D.

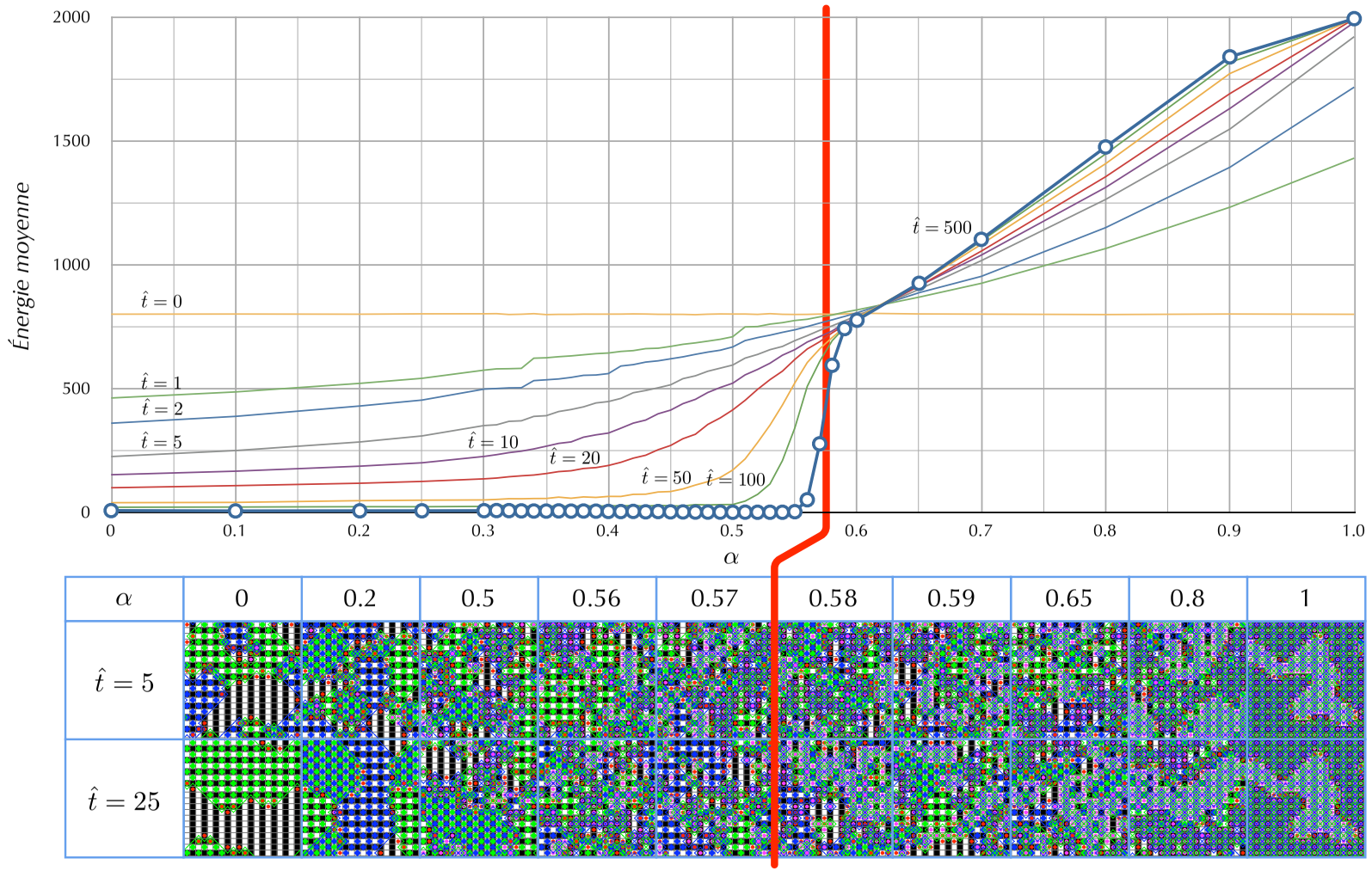


Figure 2.24 – Évolution de l'énergie moyenne en fonction de $\hat{t}$ et de α depuis 1 000 configurations aléatoires 20×20 pour M-Minorité 2D.

Remarquons que bien que les dynamiques soient différentes en physique statistique (où tout est réversible), le même type de transition de phase ordre/désordre est observé en fonction de la température dans les matériaux antiferromagnétiques (p. ex. le chrome) autour d'une température critique T_N appelée *température de Néel* ($T_N = 38°C$ pour le chrome) [316] : en dessous de T_N, les spins ont, suivant la topologie du cristal, tendance à créer des rayures où les spins sont alternivement orientés vers le bas puis et vers le haut, alors qu'au-dessus de T_N, les spins ont tous une orientation aléatoire (on parle de phase *paramagnétique*).

Étude approfondie de la stabilité des configurations en fonction de α (fig. 2.25 et 2.26). Nous savons que Minorité 2D simule Minorité 1D **ABDEGH**(**23**) à la frontière de deux bandes de damiers opposés pour vN-Minorité 2D (v. fig. 2.25(5)) ou de deux bandes diagonales rayées horizontalement et verticalement pour M-Minorité 2D (v. fig. 2.26(5)). Minorité 1D, **ABDEGH**(**23**), est le dual de l'automate **BCDEFG**(**178**) par la symétrie de damier.[34] Les simulations de **BCDEFG**(**178**) (section 3.2) démontrent donc que **ABDEGH**(**23**) admet une transition de phase sur son temps de relaxation : polynomial pour $\alpha < \alpha_c^{\mathbf{178}} \approx 0.5$ et exponentiel pour $\alpha > \alpha_c^{\mathbf{178}}$. Nous nous attendons donc à retrouver ce comportement pour $0.5 < \alpha < \alpha_c$ pour vN- et M-Minorité 2D. Or il est invisible sur les courbes d'énergie des figures 2.23 et 2.24 car les configurations concernées sont de faible énergie $O(n+m) \ll N$. Aussi, nous avons profité de l'écriture de ce manuscrit pour mener une étude complémentaire de la stabilité des configurations à $\hat{t}_{\max}$. Les figures 2.25 et 2.26 proposent trois graphiques supplémentaires pour chaque automate :

- Graphique (1) : le pourcentage des trajectoires issues de 1 000 configurations initiales aléatoires qui sont encore instables à $\hat{t}_{\max} = 500$ (notre seuil polynomial/exponentiel pour $n = 20$) ;
- Graphique (2) : le nombre moyen de cellules actives à $\hat{t}_{\max} = 500$ où la moyenne est prise uniquement sur les configurations encore instables à $\hat{t}_{\max}$ parmi les 1 000 testées pour chaque valeur de α (la moyenne est nulle par défaut, si toutes les configurations testées sont stables à $\hat{t}_{\max}$).
- Graphique (3) : les pourcentages de configurations qui à $\hat{t}_{\max}$, sont stables et d'énergie nulle (en jaune), stables mais d'énergie non-nulle (en vert), ou instables (en bleu) ;

Ces nouvelles simulations ont révélé de nouveaux phénomènes très intrigants.

Pour vN-Minorité 2D, on constate sur la figure 2.25(3) que, conformément à nos attentes, à partir de $\alpha_c^{\mathbf{178}} \approx 0.5$ une part de plus en plus importante des configurations stables d'énergie non-nulle (à deux bandes ou plus) donne lieu à un convergence lente (exponentielle), due à la transition de phase de Minorité 1D à la frontière entre les deux bandes de damiers opposés. On constate cependant que cette part chute brutalement avec la part de configurations stables d'énergie non-nulle quand α franchit la valeur 0.7, tandis que la proportion de configurations stables d'énergie nulle à $\hat{t}_{\max}$ augmente jusqu'à 93.4% pour $\alpha = 0.81$ juste avant de franchir le seuil $\alpha_c^{\text{vN}} \approx 0.825$ après lequel l'immense majorité des configurations sont instables à $\hat{t}_{\max}$ et convergent en temps présumé exponentiel. Une explication possible pour cette chute est que le motif en damier devient à partir de $\alpha \approx 0.7$ suffisamment instable pour déstabiliser l'obtention de configuration stable à plusieurs bandes de damiers, mais pas encore suffisamment pour disparaître, favorisant ainsi la convergence vers des configurations stables d'énergie nulle. Des études complémentaires à définir seront en tout cas nécessaires pour comprendre

[34] La symétrie de damier est l'opération $c \mapsto c \text{ XOR } (01)^\infty$.

ces deux nouvelles transitions. Les simulations (fig. 2.25(4)) que nous avons effectuées pour les valeurs $\alpha \in \{0, 0.25, 0.5\}$ démontrent en tout cas que les proportions de configurations stables d'énergie nulle et non-nulle restent sensiblement les mêmes quand n augmente de 10 à 100 (notons que le point $n = 100$ de cette courbe a nécessité à lui seul 3 jours de calculs intensifs, le temps nécessaire pour tester les *vingt* trajectoires aléatoires seulement !). Si l'on regarde maintenant le nombre moyen de cellules actives parmi les configurations stables à $\hat{t}_{\max}$ (fig. 2.25(2)), on constate que cette courbe est en revanche parfaitement régulière : nulle avant le seuil $\alpha_c^{\mathbf{178}} \approx 0.5$, puis linéairement croissante de faible pente jusqu'à α_c^{vN}, puis effectue un saut brutal au franchissement de ce seuil. Il semble donc que ces phénomènes présentent une certaine continuité de comportement.

La situation est plus étonnante encore pour M-Minorité 2D. On constate sur la figure 2.26(3) que le nombre de configurations stables d'énergie non-nulle commence à diminuer dès $\alpha \approx 0.3$ et que parmi ces configurations une part croissante d'entre elles voit leur temps de convergence considérablement s'allonger jusqu'à ce qu'à $\alpha \approx 0.5$ toutes les trajectoires convergent vers des configurations stables d'énergie nulle et ce, jusqu'au seuil $\alpha_c^{\mathrm{M}} \approx 0.57$ à partir duquel la quasi-totalité des trajectoires deviennent instables à $\hat{t}_{\max}$. Nous ignorons d'où provient le seuil 0.3, ni pourquoi les configurations stables semblent être toutes d'énergie nulle entre 0.5 et 0.57. Des simulations complémentaires "à la main" semblent indiquer que les configurations d'énergie non-nulle dont le temps de convergence s'allonge considérablement seraient celles composées de deux bandes rayées de la même façon mais de parité opposée (v. figure 2.26(4)). Vu la structure de ces configurations, l'hyptohèse la plus raisonnable est que le temps de convergence de ces configurations s'allonge juste d'un facteur polynomial et non exponentiel. Une hypothèse serait que les configurations toriques 20×20 soient trop étroites pour permettre l'apparition des configurations toriques permettant la simulation de l'automate Minorité 1D (figure 2.26(4)) mais ceci semble démenti par les études complémentaires[35] que nous avons menées pour les configurations de taille 30×30 et 40×40 où l'on observe ce même phénomène d'accélération de la stabilisation vers des configurations stables d'énergie nulle pour $0.5 < \alpha < 0.57$. Une autre hypothèse serait que la probabilité de générer des configurations stables à deux bandes diagonales horizontale et verticale soit constante (indépendante de n) mais nettement inférieure à 0.1% et donc invisible sur un ensemble de $1\,000$ trajectoires. Il semble en tout cas intéressant d'approfondir cette étude expérimentale pour clarifier la situation. Notons qu'en ce qui concerne le nombre moyen de cellules actives parmi les configurations instables à $\hat{t}_{\max}$, la situation semble de nouveau plus continue puisqu'à l'exception de la zone $0.5 < \alpha < 0.57$ où le nombre de configurations instables $\hat{t}_{\max}$ se raréfie, la courbe semble suivre une trajectoire croissante parfaitement lisse.

Ces simulations supplémentaires ont en tout cas révélé un fait qui me semble original : il semble que le contraste soit renforcé autour de la transition de phase "principale" (celle où l'énergie moyenne des configurations à $\hat{t}_{\max}$ passe de $O(n+m)$ à $\Omega(nm)$). En effet, tout se passe autour de cette transition comme si seules des configurations stables d'énergie nulle étaient possibles juste avant et seules des configurations instables de forte énergie après. Ainsi, une étude expérimentale un peu rapide, basée uniquement sur l'énergie moyenne des configurations et non sa distribution p. ex., pourrait conduire l'expérimentateur à croire que seules des configurations stables d'énergie nulle apparaissent avant le seuil. On en revient aux difficultés d'effectuer des simulations intensives sur un objet que l'on ne peut observer que via la mesure d'une fonctionnelle à déterminer a priori.

[35] Absentes dans ce document.

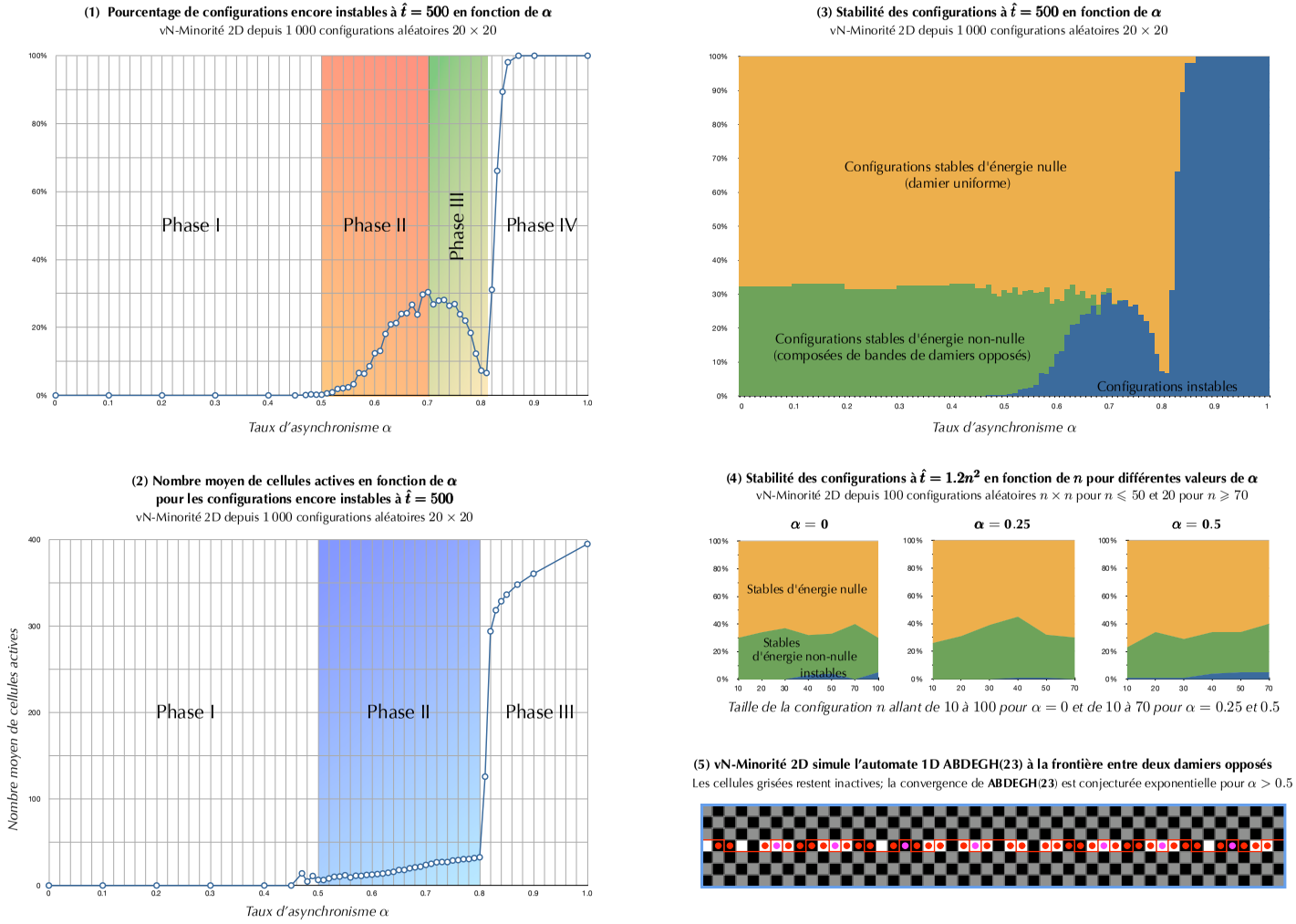


Figure 2.25 – Étude approfondie de la stabilité des configurations sur 1 000 runs en fonction de $\hat{t}$ et de α pour vN-Minorité 2D.

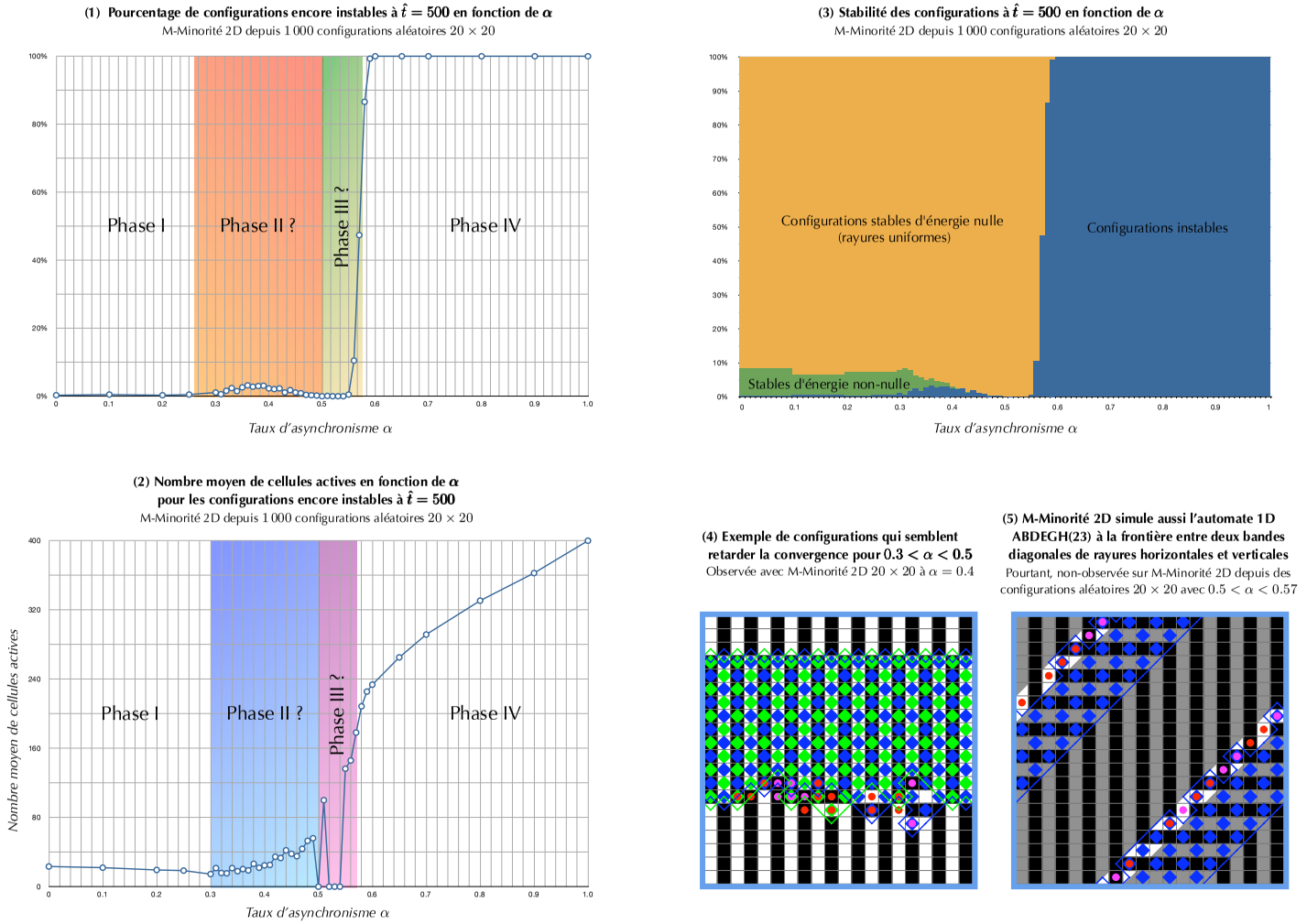


Figure 2.26 – Étude approfondie de la stabilité des configurations sur 1 000 runs en fonction de $\hat{t}$ et de α pour M-Minorité 2D.

Discussion. Notre démarche a été de mener une étude intensive sur les configurations 20×20, d'en rechercher une explication des phénomènes observés, puis de valider ces explications en menant des simulations plus ciblées pour des tailles plus grandes, dont le but est uniquement de démontrer que cette explication semble rester valable pour les plus grandes valeurs de n (cf. fig. 2.25(4)). Nous pensons que les comportements observés sont essentiellement déclenchés par des phénomènes *locaux* et devraient donc être stables quand n augmente. Nous échapperions donc au phénomène de glissement lent du seuil en fonction de n mise en évidence par Holroyd pour le phénomène hautement non-local d'envahissement de majorité par amorce [164, 149]. Il conviendrait d'évaluer précisément cette "localité", en mesurant par exemple la vitesse de propagation des nouveaux motifs dans la configuration lors des transitions. Une vitesse de propagation super-linéaire serait un indicateur convaincant de la localité du phénomène, mais le temps nous a manqué pour mener ces études complémentaires.

4.3 Analyse mathématique du régime totalement asynchrone des automates Minorité 2D

Cette section est consacrée aux résultats théoriques que nous avons obtenus sur ces automates. Ces résultats concernent uniquement le régime totalement asynchrone où à chaque pas de temps, une unique cellule choisie aléatoirement et uniformément est mise-à-jour. Le facteur de normalisation du temps est $1/N = 1/nm$. Les figures 2.27 et 2.28 présentent l'évolution de l'énergie moyenne (et de son écart-type) au cours du temps de $\hat{t} = 0$ à $\hat{t}_{\max} = 400$ calculée sur des trajectoires issues de 1 000 configurations initiales aléatoires. Ces figures représentent également l'évolution de l'énergie de quatre trajectoires typiques ainsi que, en dessous, les images des configurations d'une trajectoire typique à différents instants pour chaque automate.

On observe clairement pour les deux automates, une chute initiale brutale de l'énergie durant laquelle des motifs stables émergent un peu partout dans les configurations (des damiers pour vN-Minorité 2D et des rayures pour M-Minorité 2D). On observe ensuite une phase plus lente où l'évolution de l'énergie se fait par paliers successifs, et où de larges régions recouvertes par un motif stable entrent en compétition les unes les autres pour envahir la configuration, et s'absorbent les unes avec les autres jusqu'à ce qu'une configuration stable soit trouvée, incluant typiquement une à deux régions recouvertes chacune par un motif stable. Ce comportement est typique et l'étude précédente (fig. 2.25 et 2.26) démontre qu'expérimentalement toutes les configurations convergent vers une configuration stable en un temps court pour $\alpha = 0$. On remarquera que les bords (et diamants pour M-Minorité 2D) découpent parfaitement les configurations en régions homogènes et améliorent considérablement la lecture de l'avancement de la convergence.

On observera également que l'énergie des quelques trajectoires présentées est toujours décroissante au cours du temps, ce qui est prouvé par la proposition suivante reposant sur un simple calcul local de l'évolution des potentiels des cellules voisines de la cellule mise à jour.

Proposition 2.51 (Décroissance de l'énergie). *Sous la dynamique totalement asynchrone de vN- ou M-Minorité 2D, l'énergie est une variable aléatoire décroissante du temps. Elle varie de $8 - 4v$ chaque fois qu'une cellule de potentiel v est mise à jour. En particulier, l'énergie décroît strictement si et seulement si une cellule de potentiel $\geqslant 3$ est mise à jour.*[36]

On pourrait alors espérer que la décroissance de l'énergie suffira pour démontrer la conver-

[36] Les cellules de potentiel $\geqslant 3$ sont marquées d'un rond violet sur les figures.

gence comme pour les automates à retour positif [165, 142]. Nous verrons que l'analyse de la chute brutale initiale de l'énergie permet effectivement d'expliquer l'émergence des motifs stables au début de l'évolution, tant que la configuration n'est pas entrée dans la phase vitreuse ($E \leqslant N$). Mais nous verrons qu'ensuite, la décroissance seule de l'énergie ne suffit plus, car, comme on peut l'observer sur les quatre trajectoires typiques des figures 2.27 et 2.28, cette décroissance se fait par paliers dès que l'on est entré dans la phase vitreuse. On observera en effet qu'à partir du temps normalisé $\hat{t} = 100$ (et même 50 pour M-Minorité 2D), plus aucune cellule n'est marquée d'un rond violet dans la suite typique de configurations présentée. Cela signifie que pour la plupart des étapes, toute mise-à-jour laissera l'énergie invariante. L'apparition d'une cellule active de potentiel $\geqslant 3$ devient donc un phénomène rare et il faut introduire une autre fonctionnelle que l'énergie pour mesurer le progrès de la configuration vers une configuration stable.

4.3.1 La chute brutale initiale de l'énergie

Remarquons que toute configuration d'énergie $> 2N$ contient au moins une cellule de potentiel $\geqslant 3$ qui est mise-à-jour avec probabilité $1/N$ à chaque pas, et dont la mise-à-jour fait décroître strictement l'énergie. On en déduit que l'énergie est $\leqslant 2N$ au bout de $O(N^2)$ mises-à-jour en espérance. En étudiant une variable aléatoire mesurant l'évolution du voisinage des paires de cellules de potentiel 2 voisines, nous avons pu raffiner cette borne et obtenir le résultat suivant.

Théorème 2.52 (Chute initiale brutale [255, 258, 256, 257]). *Quelque soit la configuration initiale c^0, la variable aléatoire $T = \min\{t : E(c^t) < N + 2N/3\}$ est finie presque sûrement et $\mathbb{E}[\hat{T}] = O(N)$ pour vN- ou M-Minorité 2D en régime totalement asynchrone.*

Nous renvoyons le lecteur aux articles correspondants pour la preuve de ce théorème. Ce résultat permet d'expliquer l'émergence des motifs à damiers dans les toutes premières étapes de l'évolution totalement asynchrone de vN-Minorité 2D.

Corollaire 2.53 (Émergence des damiers pour vN-Minorité 2D [258]). *Au bout d'un temps normalisé $O(N)$, au moins $N/6$ des N carrés 2×2 de cellules sont couverts par un motif à damier dans la configuration.*

Damien Regnault a démontré ensuite dans [253] par un argument élégant reposant sur l'analyse d'une marche aléatoire tronquée mesurant l'évolution de voisinages plus grand (4×4) et une énumération assistée par ordinateur que quelque soit la configuration initiale, au bout d'un temps normalisé $O(N)$ en espérance, l'énergie passe en dessous de $N + N/8$. Il est probable qu'en étudiant des voisinages de plus en plus grands on puisse démontrer que pour tout $\varepsilon > 0$, l'énergie passe en-dessous de $(1+\varepsilon)N$ au bout de $O_\varepsilon(N^2)$ mises-à-jour aléatoires en espérance, cependant il serait intéressant d'obtenir une méthode générique.

Remarquons enfin que ces analyses ne tiennent pas compte du parallélisme de ces opérations et surestiment largement l'espérance du temps nécessaire à la chute initiale de l'énergie que nous conjecturons être polylogarithmique.

4.3.2 Évolution dans la phase vitreuse

Dès que l'énergie est inférieure à N, les arguments basés sur des statistiques locales ne donnent plus d'information sur l'avancée de la convergence car les configurations ayant ces niveaux d'énergie ne diffèrent des configurations stables qu'en un nombre négligeable de sites. Il

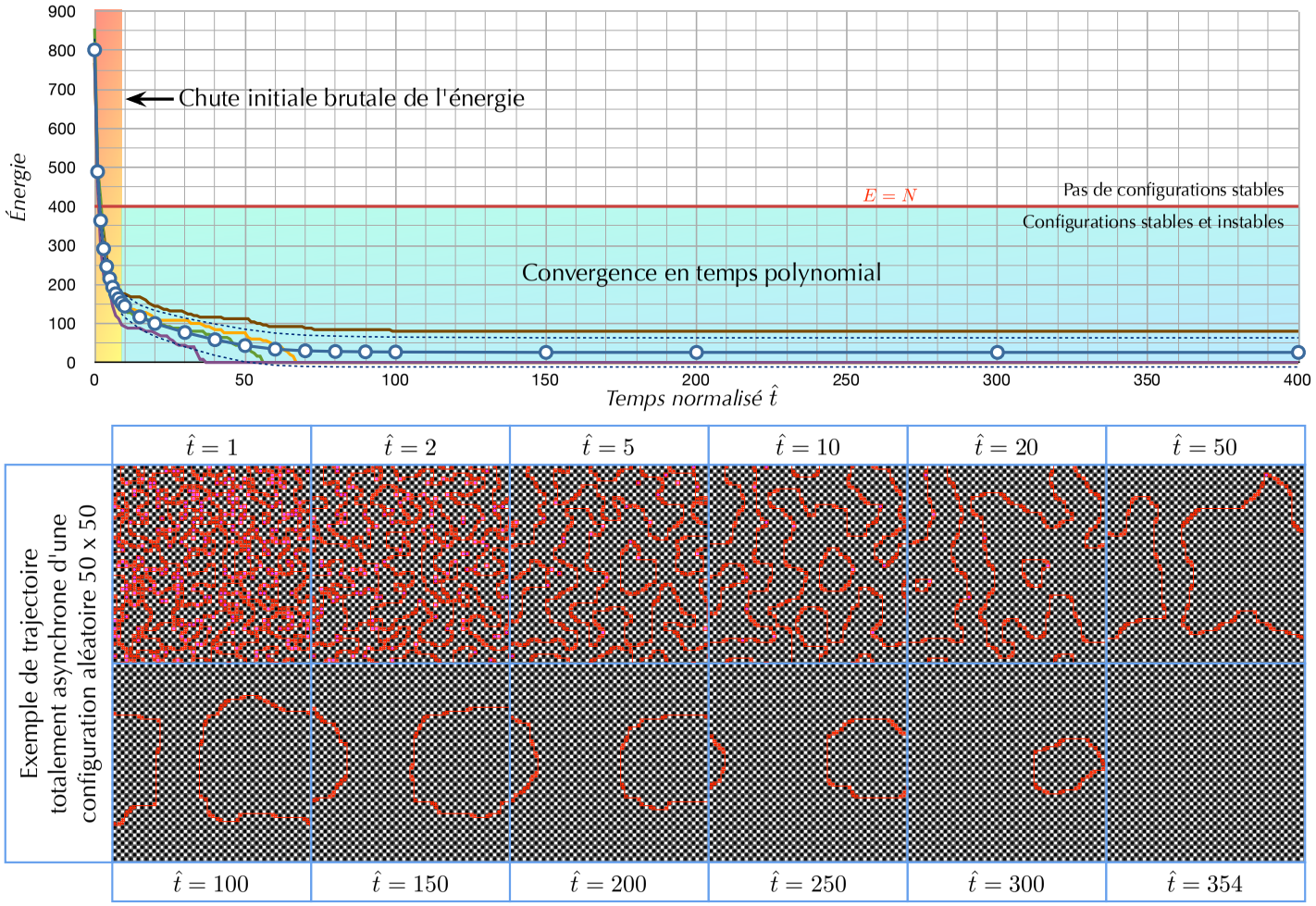


Figure 2.27 – *En haut :* Quatre trajectoires et évolution de l'énergie moyenne ± écart-type pour vN-Minorité 2D totalement asynchrone depuis 1 000 configurations aléatoires 20×20 en fonction du temps normalisé $\hat{t}$. *En bas :* Une trajectoire typique de configurations 50×50.

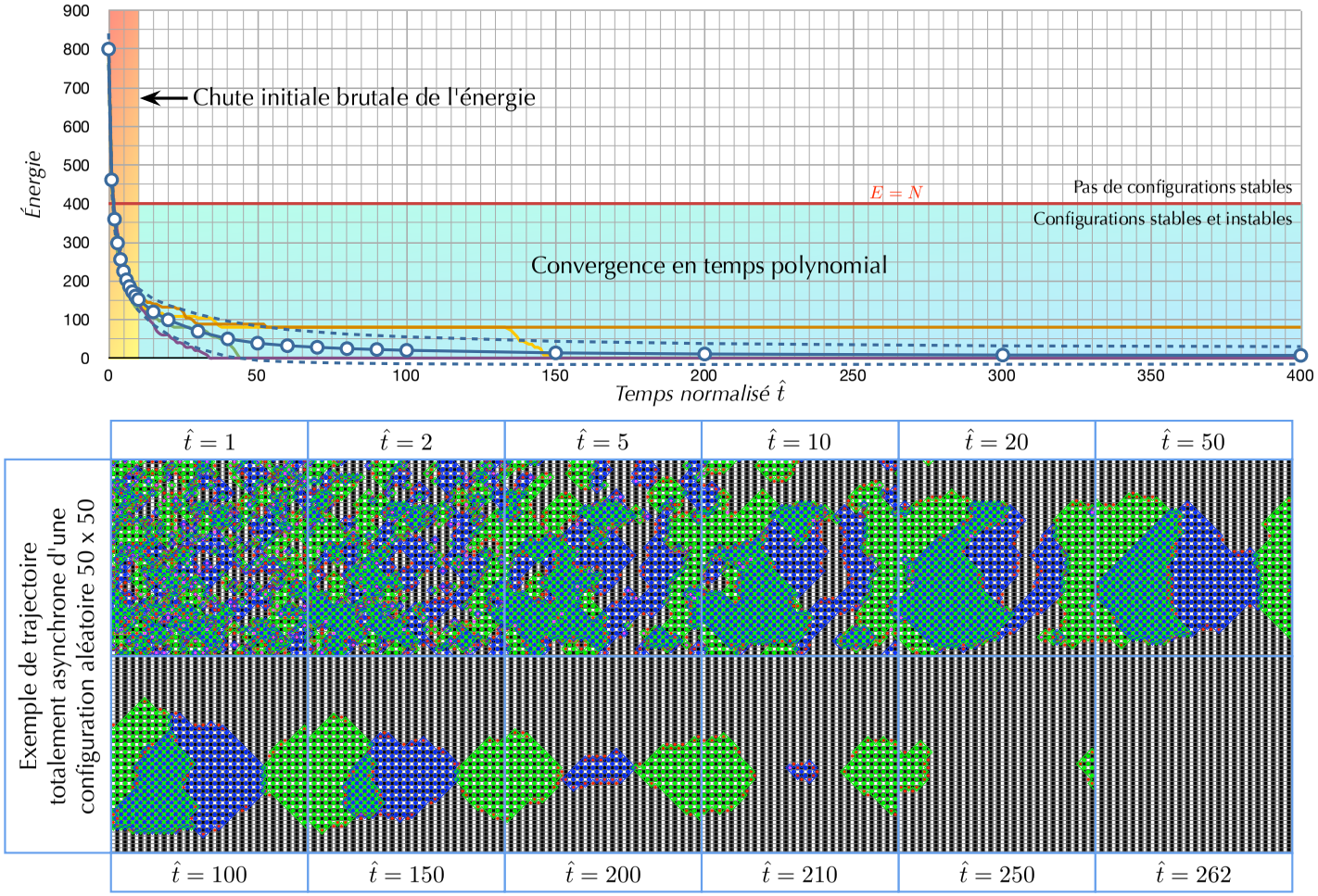
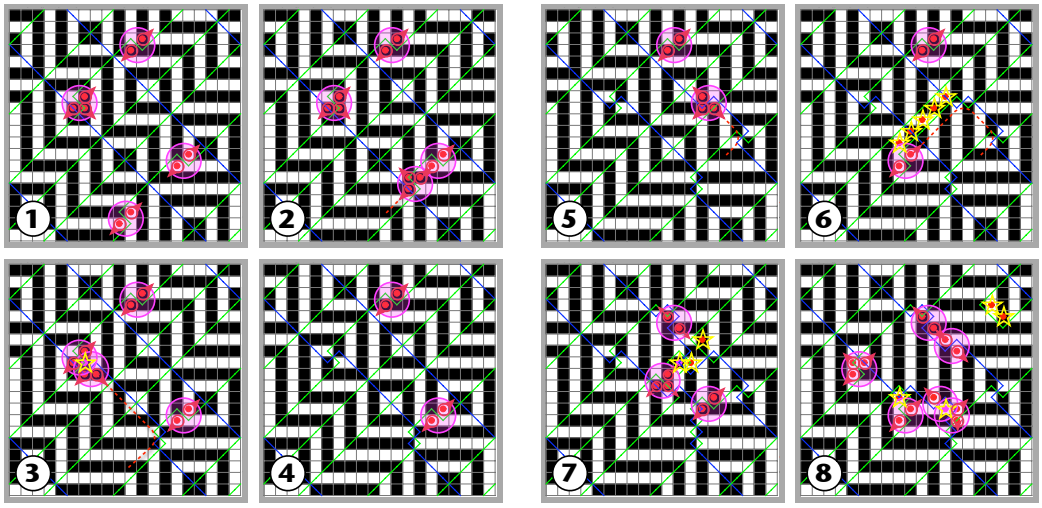


Figure 2.28 – *En haut :* Quatre trajectoires et évolution de l'énergie moyenne ± écart-type pour M-Minorité 2D totalement asynchrone depuis 1 000 configurations aléatoires 20×20 en fonction du temps normalisé $\hat{t}$. *En bas :* Une trajectoire typique de configurations 50×50.



(a) Exemple de mises-à-jour dans une configuration à quatre particules initialement où deux particules se déplacent le long des bords puis disparaissent en rentrant en collision.

(b) Exemple de mises-à-jour où une partie du réseau des bords s'effondre suite aux perturbations induites par le mouvement des particules.

Figure 2.29 – Particules d'une configuration 20×20 pour M-Minorité 2D à $\alpha = 0$.

convient alors de regarder la structure globale de la configuration. Nous ne savons pas actuellement traiter le cas général avec précision.

Convergence presque sûre. *Pour vN-Minorité 2D,* nous disposons d'une borne très lâche du temps de convergence depuis une configuration initiale arbitraire.

Théorème 2.54 ([258]). *Depuis toute configuration $n \times m$ avec n pair, vN-Minorité 2D totalement asynchrone converge presque sûrement en temps fini et après au plus $3N^{3N+1}$ mises-à-jour en espérance, et au plus $2N^{2N+1}$ si m est pair également.*

Cette borne est obtenue en exhibant pour toute configuration instable un chemin d'au plus $3N$ mises-à-jour la stabilisant. Nous conjecturons que certaines configurations $n \times m$ où n est pair et m est impair, ont un temps de convergence effectivement exponentiel en espérance (travail en cours avec Guy Fayolle et son équipe Imara, et Damien Regnault en procédant par couplage avec des processus de type KPZ), alors que nous conjecturons que l'espérance de ce temps est toujours polynomial lorsque les deux dimensions sont paires.

Pour M-Minorité 2D, la situation est plus complexe et nous ne disposons d'aucune borne sur le temps de convergence depuis une configuration initiale arbitraire. En effet, des configurations particulières (ayant une structure similaire aux points fixes de type III) permettent l'émergence de ce que nous appelons des *particules*. Il s'agit de paires ou triplets de cellules voisines de potentiel 2 (v. fig. 2.29), telles que la mise-à-jour de l'une d'entre elles déplace la particule le long des bords (dans la direction indiquée par les flèches rouges à la figure 2.29). Ces particules suivent des marches aléatoires le long des bords tout en les déformant suivant des règles très précises, et disparaissent quand elles rentrent en collision avec d'autres particules. Nous n'avons pas trouvé d'algorithme permettant de garantir leur disparition au bout d'un temps borné uniformément. La question de la convergence presque sûre de cet automate depuis toute configuration initiale est donc encore ouverte.

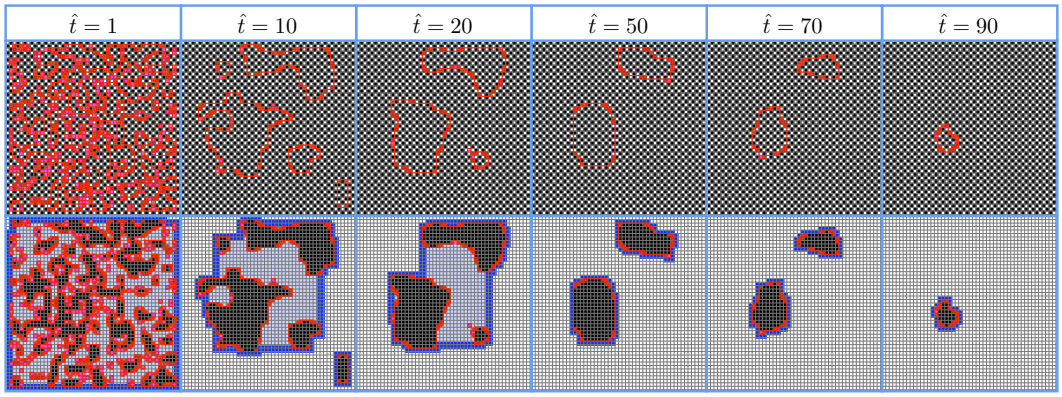


Figure 2.30 – Évolution conjointe d'une configuration 50×50 aléatoire sous vN-Minorité 2D ($\alpha = 0$), de son automate dual **OTN976**, et de son enveloppe convexe (en bleu).

Convergence depuis des configurations bornées ou semi-bornées. Nous ne savons pas comment mesurer la progression de la configuration vers un état stable lorsque les parties instables de la configuration "font le tour" du tore. En revanche, lorsque leur extension est bornée, nous pouvons évaluer sa vitesse de convergence.

Pour vN-Minorité 2D, nous dirons qu'une configuration c est *semi-bornée* (resp. *bornée*) si c est traversée par une colonne (resp. et une ligne) d'épaisseur 2 recouverte(s) d'un motif en damier. Les cellules de cette colonne (resp., de ces colonne et ligne) resteront à jamais inactives. Nous sommes alors capables d'évaluer au bout de combien de temps ce damier aura complètement envahi la configuration. Les configurations bornées sont typiques des dernières étapes de la convergence depuis une configuration aléatoire. On les observe p. ex. à partir de $\hat{t} = 50$ sur la figure 2.27.

Théorème 2.55 ([258]). *vN-Minorité 2D en dynamique totalement asynchrone converge presque sûrement en temps fini depuis toute configuration initiale semi-bornée et au bout d'un temps normalisé $O(nN)$ en espérance. Si la configuration initiale est bornée, l'espérance du temps de convergence normalisé est $O(N)$.*

Pour démontrer ce théorème, nous préférons analyser l'automate dual **OTN976** obtenu en appliquant la symétrie par damier $c \mapsto (c \,\textsc{xor}\, \boxplus)$ à la configuration : les cellules en accord avec le damier encadrant la configuration deviennent blanches, tandis que les autres sont noires.

Dans le cas de configurations semi-bornées, nous démontrons que la colonne suivant immédiatement la colonne blanche d'épaisseur 2 de la configuration duale devient elle aussi toute blanche au bout d'un temps normalisé $O(n^2)$ en espérance. Il s'en suit que la configuration duale devient toute blanche au bout d'un temps normalisé $O(n^2 m) = O(nN)$ en espérance.

Dans le cas d'une configuration bornée, nous démontrons une borne plus précise, en étudiant l'enveloppe HV-convexe[37] des cellules noires de la configuration duale. La figure 2.30 présente l'évolution typique d'une configuration bornée, de sa configuration duale et de son

[37] Un ensemble S de points de $\mathbb{Z}^2$ est *HV-convexe* si pour toute paire de points (x, y) et (u, v) de S placés sur une même ligne ou une même colonne, la totalité du segment discret $[(x, y), (u, v)] \cap \mathbb{Z}^2$ est également incluse dans S. L'*enveloppe HV-convexe* d'un sous-ensemble de points de $\mathbb{Z}^2$ est le plus petit ensemble HV-convexe qui le contienne. On notera que l'enveloppe HV-convexe n'est pas nécessairement connexe.

enveloppe HV-convexe. Le principe de la preuve est de démontrer que l'enveloppe HV-convexe évolue suivant une règle proche de celle de l'automate **OTN976** où l'on interdit les transitions qui déconnectent la configuration. On démontre alors qu'une combinaison de l'énergie de l'enveloppe convexe de la configuration et son aire, $\Phi(c) = E(h(c))/4 + A(h(c))$ où $h(c)$ désigne la configuration primale correspondant à l'enveloppe convexe de la configuration duale, décroît de $1/N$ en espérance à chaque mise-à-jour. Il s'en suit que la configuration duale, et donc la configuration primale, converge au bout d'au plus $O(AN)$ mises-à-jour en espérance où A est l'aire de l'enveloppe convexe de la configuration duale initiale, c.-à-d. au bout d'un temps normalisé $O(A)$ en espérance. Cette preuve utilise à plusieurs reprises des arguments géométriques et donc la structure globale de la configuration. Nous renvoyons le lecteur à [258] pour les détails de la démonstration.

Pour M-Minorité 2D, nous avons pu démontrer la convergence polynomiale pour les trajectoires issues de configurations constituées d'un losange de rayures horizontales placé sur un fond de rayures verticales paires. Ces configurations initiales sont dites *standard*. Par définition des diamants (v. définition 2.46 page 121), les cellules en désaccord avec le fond rayé verticalement sont celles marquées d'un diamant. Il s'agit alors de démontrer qu'au bout d'un temps polynomial en espérance, tous les diamants ont disparu. En étudiant les différentes cellules actives possibles dans une telle configuration, nous avons démontré que les configurations issues des configurations standard conservent au cours du temps une structure combinatoire récursive particulière que l'on appelle *valide*. Est valide toute configuration dont l'ensemble de diamants est :

- soit une *île*, c.-à-d. un ensemble connexe et DD'-convexe de diamants de la même couleur (où deux diamants sont voisins s'ils sont de la même couleur et sur des cellules voisines en diagonale et où D et D' désignent les deux directions diagonales) ;
- soit obtenu par juxtaposition de deux ensembles de diamants valides englobés dans des losanges R_1 et R_2 suivant l'un des schémas de la figure 2.31.

Une configuration *valide* se représente donc par son arbre de construction (v. fig. 2.32). Cet arbre de décomposition n'est pas unique et se réorganise localement au fur et à mesure que la configuration évolue. Dans la trajectoire typique de la figure 2.28, on observe des configurations valides à partir de $\hat{t} = 210$.

La figure 2.33 donne la liste des cellules actives rencontrées dans une configuration valide à symétries près (noir/blanc et horizontale et verticale). La figure 2.34 relate la réorganisation locale de l'arbre de construction d'une configuration valide pour chaque mise-à-jour possible : toute configuration valide reste donc valide. Nous définissons alors le variant $\Phi(c) = E(c)/4 + A(c)$, combinaison de l'énergie de la configuration et de son nombre de diamants $A(c)$. Nous utilisons alors la structure combinatoire arborescente de la configuration valide pour démontrer à l'aide d'arguments géométriques que :

Proposition 2.56 ([256, 257]). *Pour toute configuration valide c,* $\mathbb{E}[\Delta\Phi(c)] \leqslant -3/N$.

D'où nous déduisons :

Théorème 2.57 ([256, 257]). *M-Minorité 2D totalement asynchrone converge presque sûrement depuis toute configuration valide et au bout d'un temps normalisé* $O(A) = O(N)$ *en espérance, où* A *est le nombre de diamants initialement présents.*

Nous renvoyons le lecteur à l'article [257] pour la preuve complète de ce théorème.

Convergence de vN- et M-Minorité 2D. Ainsi, on avons été capables d'analyser les toutes premières étapes et les dernières étapes de la convergence des régimes totalement asynchrones des automates vN- et M-Minorité 2D. Contrairement au cas de vN-Minorité 2D, nous ne savons pas démontrer que M-Minorité 2D en régime totalement asynchrone converge presque sûrement depuis toute configuration lorsque l'une des deux dimensions est paire. Car nous ne connaissons pas actuellement de stratégie pour éliminer toutes les particules au bout d'un temps fini. Cette question et celle de la convergence en temps polynomial depuis n'importe quelle configuration de dimensions paires pour les deux automates restent ouvertes.

4.4 Généralisation à la classe des automates totalisants extérieurs

Dans les simulations que nous avons menées, nous avons isolé de nombreux autres automates totalisant externes présentant des caractéristiques similaires à celle de minorité : apparition de larges régions couvertes de motifs stables qui entrent en concurrence, fusionnent jusqu'à ce qu'une configuration stable soit trouvée. Certains se comportent de façon très similaires et possèdent également des particules. Pour d'autres, les particules prennent l'aspect de nuages de bruit aléatoire (p. ex. l'automate **OTM244** [282]) et semblent présenter des défis encore plus importants pour l'analyse.

Concernant les régimes α-asynchrones, nous ne disposons actuellement d'aucune méthode pour les traiter. Les dernières simulations que nous avons menées (relatées à la section 4.2) laissent penser que ces automates traversent entre trois et quatre phases successives où les motifs stables perdent progressivement en stabilité au fur et à mesure que α augmente. Il est certain qu'une étape importante dans la compréhension de ces transitions sera de comprendre les transitions de phase observées dans les régimes α-asynchrone de l'automate 1D **ABDEGH**(**23**) ou encore de son automate dual **BCDEFG**(**178**).

5 Travaux postérieurs et perspectives

Retombés de nos travaux. Nous avons présenté nos travaux à différentes conférences ne relevant pas directement des systèmes dits complexes : depuis 2003 ces travaux ont fait l'objet de présentations régulières à l'indispensable école ALÉA qui se tient chaque année au CIRM, ainsi qu'à d'autres séminaires moins spécialisés comme le séminaire de Dagstuhl sur l'approximation et les algorithmes randomisés en 2005, où nous avions alors présenté nos travaux sous l'angle de l'influence de l'introduction du hasard dans les calculs [280]. L'accueil à Dagstuhl de ces résultats avait été étonnamment bon, réveillant dans une partie de l'auditoire de nombreuses questions posées dans les années 1960-1970 sur les systèmes dynamiques discrets (les vers de Paterson p. ex. [320]). Nous avons eu l'agréable surprise de découvrir à la suite de ces séminaires que les phénomènes intrigants que nous avons mis en évidence tout d'abord avec Nazim Fatès et Éric Thierry puis avec Damien Regnault, ont eu un effet d'entraînement sur une partie de la communauté. Des chercheurs plus aguerris que nous en théorie des probabilités, nous ont emboîté de le pas pour approfondir nos résultats et apporter des précisions sur nos résultats très primitifs à l'époque. Parmi eux, Philippe Chassaing et Lucas Gérin (alors en thèse sous sa direction) ont étendu nos résultats sur l'espérance du temps de relaxation en régime totalement asynchrone des automates cellulaires élémentaires, pour démontrer les convergences en loi de ces processus [63]. Leurs résultats ont confirmé la stricte identité de ces processus avec des marches aléatoires. Lucas Gérin a par la suite obtenu d'autres résultats avec

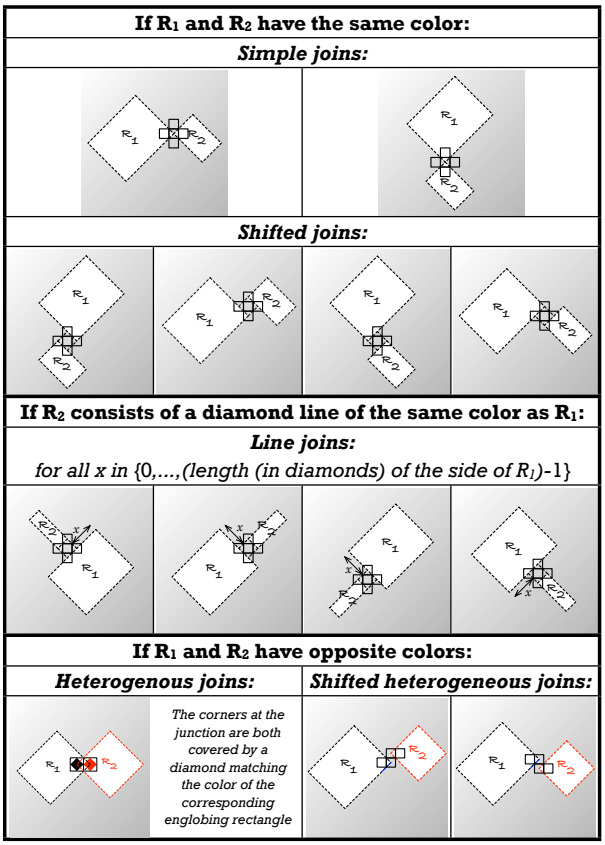


Figure 2.31 – Définition récursive des configurations valides (M-Minorité 2Dà $\alpha = 0$).

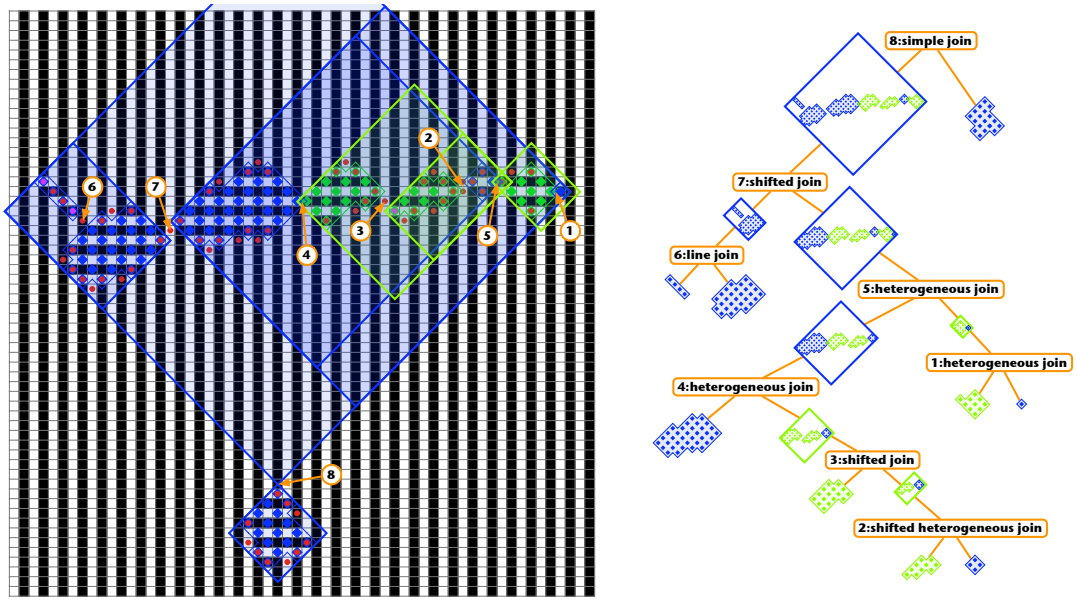


Figure 2.32 – Un exemple de configuration valide, avec ses losanges englobants et son arbre de décomposition. Remarquons que les losanges englobants utilisés dans deux étapes consécutives de la construction ne sont pas forcément inclus les uns dans les autres, v. les étapes 4 et 5 de cette décomposition.

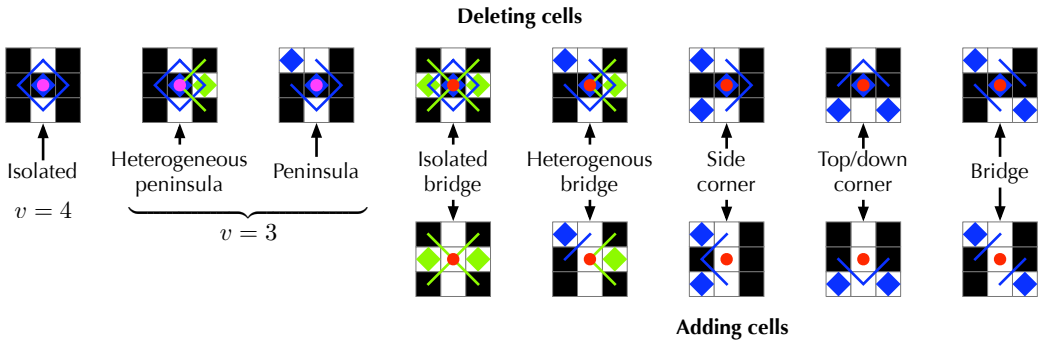


Figure 2.33 – Les cellules actives (après symétries) d'une configuration valide (M-Minorité 2D).

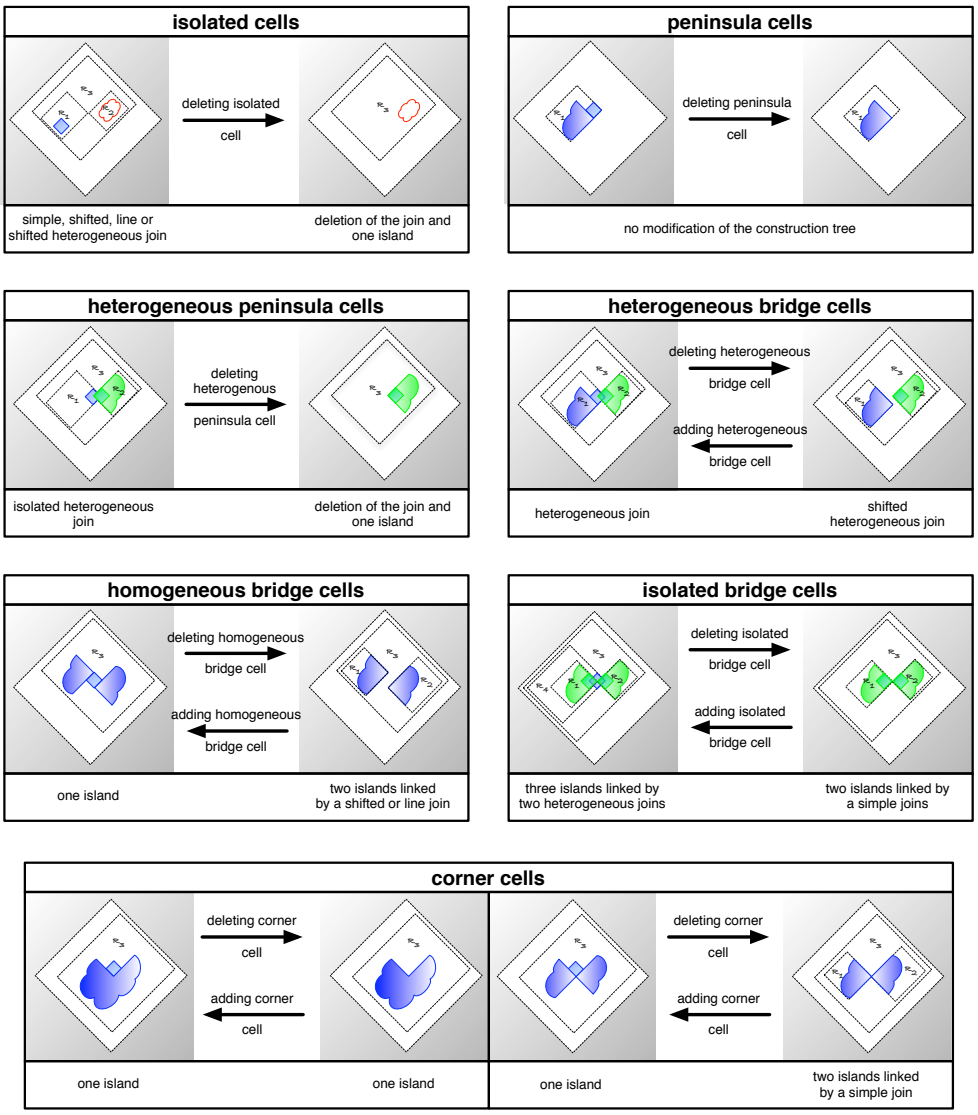


Figure 2.34 – Une configuration valide donne une configuration valide après chaque mise-à-jour de M-Minorité 2Dà $\alpha = 0$.

Nazim Fatès sur des automates cellulaires totalisants 2D pour le voisinage de von Neumann [108].

Les recherches sur les automates cellulaires en théorie des probabilités en mathématique sont historiquement divisées en deux communautés assez étanches : les particules en interactions qui s'intéressent au cas où chaque cellule dispose de sa propre horloge pour se mettre à jour (correpondant à notre régime totalement asynchrone), et les automates probabilistes, similaires aux régimes α-asynchrones avec $0 < \alpha < 1$; les deux travaillant sur des configurations bi-infinies. Notre modèle propose peut-être un cadre naturel pour créer des liens entre les résultats de ces trois champs de recherches comme le suggère Bernard Ycart dans son rapport sur la thèse de Damien Regnault. À la suite des discussions que nous avons eu avec Jean Mairesse depuis que je suis arrivé au LIAFA, nous nous sommes convaincus que l'objet d'étude principal de ces domaines, à savoir les distributions limites et en particulier décider de leur unicité en fonction de différents paramètres, est intimement lié au paramètre algorithmique que nous avons étudié dans ce manuscrit : le temps de relaxation. Jean Mairesse et Irène Marcovici [205] ont par exemple utilisé le résultat de la thèse de Damien Regnault [252] exhibant un couplage entre **BCDEFG**(**178**) et la percolation dirigée pour démontrer l'existence d'une mesure invariante non-triviale pour certaines valeurs du paramètre α pour l'automate **BCDEFG**(**178**). Nous explorons actuellement certaines hypothèses avec eux pour exhiber des situations où les transitions de phase du temps de relaxation polynomial/exponentiel soient mathématiquement équivalentes à l'apparition de mesures invariantes non-triviales pour ces processus. Même si cela ne permettra peut-être pas de démontrer l'existence de ces transitions, nous aurions alors à notre disposition un outil bien plus simple à simuler pour les étudier.

Nous allons par ailleurs prendre part cette année au comité de programme de la première édition de la conférence internationale sur les automates cellulaires asynchrones, un satellite de la conférence ACRI, qui permettra de mieux structurer la communauté autour de ces questions.

Perspectives. Le point de départ de nos travaux était l'étude de l'adéquation entre les phénomènes observables par simulation et la réalité des comportements des automates cellulaires. Nos travaux s'étaient donc naturellement portés sur l'étude du temps de convergence de ces automates : un temps de convergence polynomial offrant la garantie que l'on puisse l'observer par simulations. L'originalité de notre approche est le choix d'une exploration exhaustive de l'ensemble des comportements possibles des petits automates cellulaires en régime asynchrone. Notre projet à long terme est en effet d'identifier les comportements des petits automates afin de recenser d'autres façons plus économiques d'effectuer certains calculs. Nous sommes en effet convaincus que nous ignorons encore de très nombreux mécanismes économiques en ressources pour mener des calculs. Il nous semble peu probable que les mécanismes utilisés par la nature soient similaires à ceux que l'on conçoit habituellement en informatique. Les solutions les plus efficaces connues[38] sont extrêmement structurées, peu robustes aux erreurs et le plus souvent inefficaces en terme de ratio *qualité* du résultat/ressources monopolisées pour son calcul. Nous l'avons vu dans l'introduction, avec les compteurs aléatoires p. ex. : il existe bien souvent des alternatives randomisées bien plus compactes et qui même si elle ne produisent pas toujours le résultat exact, en produisent un avec une précision suffisante à la plupart des applications en consommant exponentiellement moins de ressources. Cette recherche se situe à l'intersection d'au moins quatre domaines : théorie des probabilités, automates cellulaires, simulations, et algorithmes d'approximation et randomisés.

[38] Il est d'ailleurs amusant de constater que l'optimisation du tri d'entiers est toujours un sujet brûlant, v. [157].

3 Ordonnancement à l'aveugle

1 Introduction

Nous avons vu dans les deux chapitres précédents deux aspects du champ des systèmes complexes : la modélisation avec l'étude des petits mondes et l'analyse systématique des modèles avec les automates cellulaires probabilistes. Nous nous intéresserons ici à un troisième aspect : l'absence de données réellement fiables sur les phénomènes que l'on cherche à contrôler.

Faire face à l'imprévu. La situation la plus standard dès que l'on s'attaque à une situation réelle est l'absence de données ou de connaissances en suffisance. Dans la plupart des situations, les données sont découvertes au-vol, et les problèmes au moment où ils se posent : p. ex., le retard pris suite à la découverte d'un site archéologique unique en creusant un tunnel de métro. Cette incertitude permanente sur la visibilité que l'on a de la conduite d'un projet induit des phénomènes difficiles à contrôler sur le long terme. La plupart des techniques connues ne peuvent d'ailleurs garantir des délais que sous la réserve qu'aucun imprévu ne se produise, et n'offrent aucune garantie sur la qualité de la gestion en cas d'imprévu. En témoigne la grande difficulté à budgétiser la plupart des grands projets. La multiplication des corps de métiers et des centres de réalisation impliqués dans un même projet tend à aggraver considérablement le nombre d'imprévus dans la conduite des gros projets industriels : p. ex., les avions sont maintenant conçus, construits, assemblés et testés sur des sites tous différents et ayant chacun la responsabilité d'une partie seulement du projet, et maintenir une vision globale du projet devient très délicat ; en témoigne la mésaventure de l'airbus A380 où des sites ayant la responsabilité de sections différentes de l'avion avaient placé les câblages à des positions différentes et incompatibles. Toutes ces incertitudes font qu'il est même difficile de prévoir correctement les interdépendances. Le désir de rechercher des moyens de reprendre la main face à tous ces imprévus a donné naissance récemment à un nouveau champ de recherche dits des *systèmes complexes industriels*. Une des ambitions principales de ce domaine est de permettre et de contrôler l'*émergence* d'un ordre (ou d'une structure) dans ces projets titanesques par le moyen de directives *locales légères*, garantes à la fois du bon déroulement du projet mais aussi de la maîtrise de son coût.

L'objet de cette section est de démontrer qu'il est effectivement possible de concevoir des

algorithmes d'ordonnancement capables de s'auto-réguler et d'offrir des garanties de performances dans des situations très générales incluant des contraintes de précédence arbitraires, sans aucune visibilité sur ce qu'ils ont à faire, ni même de ce qu'ils font, ou ont fait.

Maîtriser les coûts. La première chose qui a frappé les chercheurs (Graham, Johnson, etc.) qui ont défriché le domaine de l'ordonnancement dans les années 1960-70, est l'incroyable instabilité des algorithmes : le moindre changement de longueur d'une tâche, du nombre de processeurs, etc. peut induire des comportements complètement inattendus même chez les algorithmes les plus simples. Un exemple typique est la non-monotonie surprenante de l'algorithme **First-Fit** en *bin-packing* :[39] il existe des listes d'objets telles que si vous ôtez un des objets (souvent le plus petit, en croyant bien faire), **First-Fit** utilise une boîte de plus. Ces "anomalies" ont fait l'objet de très nombreux papiers, p. ex. [144, 145, 146, 147]. Elles compliquent considérablement les preuves : la complexité d'apparence démesurée des preuves des performances de **First-Fit** tient beaucoup à sa non-monotonie [177, 178]. La preuve de la NP-complétude de ces problèmes a permis de se focaliser sur l'approximation du coût des solutions. Le but devient alors de rechercher une solution dont on puisse garantir que son coût n'est jamais très éloigné du coût optimal. Ceci a permis dans la plupart des cas de lisser un peu les difficultés (v. l'étude des algorithmes de listes [148] pour quelques uns des tous premiers résultats d'approximation). Depuis le milieu des années 1970, ont ainsi vu le jour de nombreux algorithmes dits *d'approximation* ou c-approximations, au sens qu'ils garantissent de produire une solution dont le coût est toujours à un facteur $\leqslant c$ du coût optimal (v. [308, 309] pour un état de l'art des techniques utilisées sur des exemples variés). Dans le même temps, durant les années 1975-1995, l'apport de la complexité randomisée et la caractérisation de la classe NP en terme de preuves vérifiables stochastiquement ($NP = PCP(\log n, 1)$) ont permis de raffiner la classification des problèmes en les hiérarchisant par leur (in)approximabilité [18]. On dispose maintenant de techniques qui démontrent par de simples réductions qu'à moins que $P = NP$, tel problème n'admet pas de $f(n)$-approximation où $f(n)$ peut-être un polynôme, une logarithme, une constante, etc. En ce sens le meilleur algorithme dont on puisse rêver pour un problème NP-complet est un schéma d'approximation (A_ε) qui pour tout $\varepsilon > 0$, calcule une solution en temps polynomial (fonction d'ε) une solution de coût quasi-optimal $\leqslant (1+\varepsilon)\mathrm{OPT}$. Ce faisant, pour obtenir ces résultats, les algorithmes ont dû gagner en complexité et leur mise en œuvre effective devient parfois très délicate (ne serait-ce que leur implémentation à des fins expérimentales).

Faire face à un futur incertain. Parallèlement à la recherche d'algorithmes de plus en plus précis et complexes, disposant d'informations très complètes sur l'instance étudiée, s'est développé un champ de recherches pour essayer de valider et d'améliorer les approches adoptées en pratique où l'on ne dispose pas de toutes ces informations ni de beaucoup de temps de calcul pour prendre ses décisions. Depuis le milieu des années 1970, l'algorithmique *en-ligne* étudie le cadre où l'algorithme ne découvre les requêtes qu'au fur et à mesure de leur arrivée, v. [51]. L'incertitude du futur se révèle être une contrainte très forte puisque dans la plupart des cas... *il n'existe pas de stratégie optimale* proprement dite et ce dans l'absolu, sans la moindre hypothèse de complexité du type $P \neq NP$ ou autre, décidable ou non ! En

[39] Étant donnés n objets unidimensionels, le problème du *binpacking* consiste à placer ces objets dans un nombre minimum de boîtes de taille 1. L'algorithme **First-Fit** consiste à numéroter de 1 à n toutes les boîtes et à placer l'objet suivant dans la première boîte de 1 à n où il tient. On constate alors [226] que **First-Fit** utilise trois boîtes pour ranger la suite d'objets $.50, .67, .51, (.2), .49, 0.24, 0.33, 0.24$ et quatre si on supprime le plus petit objet, $.2$! Les problèmes de *bin packing* sont souvent étudiés conjointement avec les problèmes d'ordonnancement car ils sont dual l'un de l'autre : les boîtes correspondant aux processeurs et leur taille à la durée de l'ordonnancement.

effet, comme l'instance est révélée à l'algorithme au fur et à mesure de son exécution, on peut considérer qu'elle est construite par un adversaire tout puissant qui décide le futur de l'instance en fonction des décisions prises par l'algorithme pour l'amener aussi loin qu'il le peut de la solution optimale *hors-ligne* (c.-à-d., connaissant la totalité de l'instance dès le départ). Par exemple, quelque soit l'algorithme / la stratégie / les choix en-ligne que l'on fait, il est toujours possible de construire une instance de *bin-packing* dont la solution engendrée aura un coût $\geqslant (1.5 - \varepsilon)\,\mathrm{OPT}$ pour tout $\varepsilon > 0$ [329]. S'il n'existe pas de stratégie optimale, comment évaluer les performances des algorithmes en-ligne ? Le consensus actuel est de comparer le coût de la solution qu'il engendre face à un adversaire tout puissant qui construit l'instance en fonction des choix effectués, au coût optimal hors-ligne. On parle d'algorithmes *c-compétitif* lorsque le coût de la solution produite face à cet adversaire est toujours à un facteur $\leqslant c$ du coût optimal hors-ligne. Différents types d'adversaires peuvent être envisagés pour raffiner cette définition, v. [51].

Analyse par augmentation de ressource. Pour certains problèmes, comme celui de la gestion du cache mémoire [331], tous les algorithmes ont le même facteur de compétitivité alors que certains sont clairement plus efficaces que d'autres en pratique. Il a alors été proposé dans [292] de raffiner la notion d'analyse compétitive en limitant la quantité de ressources dont dispose la solution optimale hors-ligne. Le coût de la solution produite par l'algorithme en-ligne sur un cache de taille n est alors comparé au coût de la solution optimale sur un cache de taille $k < n$. La dépendance du facteur de compétitivité en fonction du rapport n/k indique comment l'algorithme gère la réduction de sa marge de manœuvre et a permis de mettre en valeur les qualités respectives des différents algorithmes. Pour de nombreux problèmes d'ordonnancement, aucun algorithme en-ligne ne peut approcher la solution optimale hors-ligne à aucun facteur constant, v. [245]. Aussi, [180, 238] proposèrent d'adapter ce principe au domaine de l'ordonnancement en proposant de comparer le coût de l'ordonnancement produit par l'algorithme en-ligne sur sp processeurs au coût optimal sur p processeurs pour $s > 1$. Un algorithme d'ordonnancement est dit *s-rapide c-compétitif* si son coût sur sp processeurs reste à un facteur $\leqslant c$ du coût optimum sur p processeurs pour toute instance. On interprète la s-rapide c-compétitivité d'un algorithme comme sa capacité à *supporter une charge* $1/s$: en effet, son coût reste à un facteur constant de l'optimum limité à une fraction $1/s$ des processeurs, c.-à-d. "chargé à $1/s$ %". En pratique on constate effectivement que les performances des systèmes (celles des différents protocoles de réseaux, p. ex.) s'effondrent brutalement au-delà d'une certaine charge limite, où le coût commence à croître fortement avec le nombre de requêtes, c.-à-d. diverge progressivement par rapport à l'optimum au fur et à mesure que le temps passe. Ce qui explique d'ailleurs pourquoi nous sommes parfois amenés à rebooter notre ordinateur après une charge de travail anormalement élevée pour retrouver un niveau de performance acceptable en repartant à zéro. L'objectif est alors de rechercher des algorithmes supportant la charge la plus élevée possible, c.-à-d. s-rapide $O(1)$-compétitif pour le plus petit s possible.

Nous renvoyons le lecteur aux états de l'art très complets et récents [244, 245] sur l'analyse compétitive d'ordonnanceur en-ligne.

Vers des algorithmes prouvés efficaces en pratique. Dans la plupart des situations réelles, on ne dispose d'aucune information précise sur la quantité réelle de travail que représente chaque tâche demandée au moment où elle nous est soumise (la notion de travail est d'ailleurs la plupart du temps difficile à définir [77]). La plupart des systèmes d'exploitation ordonnancent les processus qu'on leur soumet, sans rien savoir de leurs caractéristiques : ils

ignorent leur quantité de travail, leur parallélisabilité (l'accélération de son temps d'exécution sur plusieurs processeurs), le nombre de sous-processus qu'ils vont engendrer, etc. En fait, les seules données dont ils disposent dans les situations réelles, sont les dates d'arrivée des tâches et celles où ils les ont terminées. Le cadre des algorithmes en-ligne est alors inadapté, v. [299]. Il est p. ex. impossible de mettre en œuvre la stratégie en-ligne optimale pour le temps de service moyen, **SRPT** *(Shortest Remaining Processing Time)* qui ordonnance systématiquement la tâche disponible dont le travail restant est minimum, quitte à préempter[40] le processus en cours d'exécution [294]. En effet, si on ne connaît pas la durée des tâches, il est impossible de décider celle dont le travail restant est minimum.

Nous allons étudier plus précisément le *temps de service moyen*, correspondant au temps de traitement moyen des requêtes des utilisateurs, qui est une mesure naturelle des performances de ces systèmes interactifs. On pourra consulter [245] pour l'étude d'autres fonctions objectif.

Algorithmes non clairvoyants. Motwani, Philipps et Torng proposent d'étudier la minimisation du temps de service moyen en supposant que l'ordonnanceur est *non clairvoyant*, c.-à-d. qu'il découvre les tâches au fur et à mesure de leur arrivée et doit les ordonnancer sans rien connaître de leur quantité de travail a priori, et qu'il est seulement informé de leur terminaison au moment de leur terminaison et pas avant [224]. Ils autorisent la préemption des tâches, indispensable pour obtenir de performances acceptables dans un système interactif.

Ils démontrent alors que la non-clairvoyance a un coût : tout algorithme non clairvoyant est nécessairement $\Omega(\sqrt[3]{n})$-compétitif s'il est déterministe, et $\Omega(\log n)$-compétitif s'il est randomisé. Leur cadre permet également de développer des stratégies compétitives, ce qui est un premier pas vers des algorithmes réellement implémentables en pratique. [180] démontre que l'algorithme non clairvoyant **SETF** *(Shortest Elapsed Time First)*, qui tente d'approximer la stratégie **SRPT** en se basant sur le travail déjà effectué, est $(1+\varepsilon)$-rapide $(1+\frac{1}{\varepsilon})$-compétitif, c.-à-d. résiste à une charge arbitrairement proche de 100%. [181] propose une première analyse de l'algorithme non clairvoyant randomisé **RMLF** (*Randomized Multi-Level Feedback*, v. [245]) dont [36] démontrera qu'il est en fait $O(\log n)$-compétitif, c.-à-d. optimal.

La plupart de ces études considèrent toutefois que les tâches ont toute le même profil d'accélération ou ne peuvent s'exécuter que sur un seul processeur à la fois. Or, la plupart des programmes alternent entre des phases où ils ne tirent aucun avantage de plusieurs processeurs (soit que cette partie du code n'a pas été prévue pour cela ; soit que le programme est tout simplement en attente active d'une interaction) et des phases où augmenter le nombre de processeurs est utile (comme dans le cas du parallélisme de données où le même code est exécuté sur des données disjointes : p. ex., le calcul d'une image par lancers de rayons où chaque rayon peut être lancé par un processeur différent). Ce problème se pose de façon d'autant plus intense qu'avec l'avénement des ordinateurs multi-core et l'incapacité actuelle à développer des algorithmes tirant réellement parti du parallélisme (à l'exception de certaines applications, vidéo p. ex.), les systèmes d'exploitation doivent maintenant ordonnancer des tâches qui tirent avantage de façon très aléatoire et imprévisible des processeurs qu'on leur

[40] La *préemption* consiste à interrompre une tâche, pour en exécuter d'autres, puis reprendre son exécution plus tard au point où l'on s'était arrêté. La préemption est souvent indispensable pour pouvoir garantir des performances acceptables en-ligne dès que les tâches peuvent avoir des durées très différentes. Un adversaire malicieux peut en effet très facilement faire flamber le temps de service moyen de n'importe quel ordonnanceur non-préemptif en posant un nombre arbitrairement grand de requêtes pour des tâches arbitrairement petites, chaque fois que l'ordonnanceur commence à en diffuser une grande. Pour cette raison, les systèmes d'exploitations ont tous progressivement intégré la préemption à partir de la fin des années 1960.

attribue, et ce de façon très inégale au fur et à mesure de leur avancement.

Profils d'accélération arbitraires. Les travaux sur les algorithmes non clairvoyants ci-dessus ne considèrent que des tâches parallèles, qui s'exécutent x fois plus rapidement sur x fois plus de processeurs. Jeff Edmonds propose alors dans [93] d'étudier le cas où les tâches à ordonnancer traversent différentes phases avec différents de degré de parallélisme décrits par des fonctions d'accélérations arbitraires (v. section 2). Il démontre dans ce cadre très général que l'algorithme **Equi** qui partage équitablement les processeurs entre les tâches actives, est $(2+\varepsilon)$-rapide $O(1)$-compétitif, c.-à-d. peut supporter une charge arbitrairement proche de 50%. Un intérêt du modèle d'Edmonds est sa capacité à pouvoir y réduire d'autres problèmes réputés difficiles pour en obtenir des algorithmes compétitifs : [97] en déduit un premier algorithme compétitif pour la diffusion de message en-ligne par *multicast* ; et [96] en déduit la première analyse du protocole TCP en l'absence de toute hypothèse sur la distribution des requêtes.

Notre contribution : introduction des dépendances. La plupart des tâches réelles se décomposent en plusieurs sous-tâches qui doivent être exécutées en respectant certaines contraintes de précédence. La forme générale de ces contraintes est celle d'un graphe orienté acylique (DAG, en abrégé). La structure de ces contraintes peut parfois être assez légère : dans le cas d'un serveur web, le téléchargement d'une page web simple est terminé quand le téléchargement de chaque élément qui la compose est fini, quelque soit l'ordre dans lequel ils sont téléchargés ; une requête pour une page web consiste donc en un ensemble de requêtes sans relation de précédence entre elles. Suivant les applications, différentes formes de relations de précédences peuvent être envisagées.

Avec Julien Robert, nous avons exploré l'influence de ces dépendances dans le contexte des algorithmes non clairvoyants défini par Edmonds. Nous étendons le modèle de [93] en y incorporant des contraintes de précédences entre les différents processus qui composent chaque tâche. L'ordonnanceur (non clairvoyant) n'est pas informé des contraintes de précédences des tâches et découvre les différents processus de chaque tâche au fur et à mesure qu'ils apparaissent, quand tous leurs prédécesseurs sont terminés.

Nos travaux [264, 266] ont démontré que l'algorithme non clairvoyant **Equi** $\circ$ **Equi** est $(2+\varepsilon)$-rapide $O(\kappa)$-compétitif pour tout $\varepsilon > 0$, où κ est une constante qui dépend uniquement de la structure des tâches et non du nombre de requêtes. La conclusion étonnante de nos travaux est donc que la présence de dépendances entre les tâches *ne diminue pas la charge maximale que peut supporter un serveur* ! La clé de ce résultat est de répartir d'abord les ressources entre les tâches puis de répartir les ressources attribuées à chaque tâche entre ses processus. Nous démontrons en effet qu'ordonnancer les processus actifs en oubliant leur origine peut conduire à une dégradation arbitraire des performances. Notre résultat repose sur l'introduction d'un nouveau paramètre, le *coefficient d'étalement*, qui mesure la capacité d'une structure de dépendances à s'exécuter de manière efficace. L'analyse de ce coefficient, qui se calcule facilement, nous a permis d'obtenir des facteurs de compétitivité plus précis pour différentes familles classiques de dépendances.

Auparavant, nous avons étudié dans [265], le cas de l'optimisation de serveur web *multicast*. Nous avons démontré qu'il était possible de reprendre l'analyse de [97] pour l'étendre au cas de requêtes inter-dépendantes et démontré que l'algorithme **Equi** $\circ$affairé (v. section 6) est s-rapide $O(1)$-compétitif pour tout $s \geqslant 4+\varepsilon$. C'est dans ce contexte que nous avions introduit les algorithmes $A \circ B$ répartissant les ressources en deux temps que nous avons utilisés par la suite dans [264, 266] : un algorithme A qui répartit les ressources entre les requêtes,

suivi d'un algorithme B qui répartit les ressources attribuées à chaque requête (par A) entre les éléments qui composent la requête.

Une de nos contributions importantes avec le recul a été de clarifier et préciser les travaux de [93, 97] qui du fait de leur complexité (reconnue par leurs auteurs eux-mêmes, v. [94]) étaient restés confinés dans un cercle très réduit d'initiés, et n'avaient pas rencontré l'écho que l'on pouvait en espérer dans la communauté. À la suite de nos travaux, Edmonds et Pruhs ont démontré que la famille d'algorithmes $(\textsc{Laps}_\beta)_{0<\beta\leqslant 1}$ (v. section 4) est s-rapide $\frac{4s}{\beta\varepsilon}$-compétitif pour tout $s > 1+\beta+\varepsilon$, et supporte donc une charge arbitrairement proche de 100% pour β suffisamment petit. Leur analyse simplifie considérablement celle de $\textsc{Equi} = \textsc{Laps}_1$ initialement proposée dans [93]. Nous avons pu simplifier encore leur démonstration et gagner un facteur 2 sur le facteur de compétitivité en utilisant les méthodes que nous avons développées dans [265, 264, 266], et démontrer ainsi que $\textsc{Laps}_\beta$ est en fait s-rapide $\frac{2s}{\beta\varepsilon}$-compétitif pour tout $s > 1+\beta+\varepsilon$ [268, 267]. Comme nos résultats de [265, 266] procédaient de façon modulaire par rapport à [93, 97], nous avons pu les adapter sans difficulté pour utiliser $\textsc{Laps}_\beta$ plutôt qu'**Equi** dans nos analyses. Ce sont ces nouveaux résultats que nous vous présentons ici.

Plan du chapitre. La section 2 présente le modèle avec contraintes de précédence que nous étudions. La section 4 présente notre amélioration à la marge de l'analyse de [99] en l'absence de contraintes de précédence. La section 5 présente notre résultat principal, le théorème 3.8 affirmant que la famille d'algorithmes non clairvoyants $\textsc{Laps}_\beta \circ \textsc{Equi}$ est s-rapide $\frac{s(1+\kappa)}{\beta\varepsilon}$-compétitif pour toutes structures de précédence avec moins de κ processus deux-à-deux indépendants. La section 5.4 étudie le coefficient d'étalement pour obtenir de meilleurs facteurs de compétitivité pour des structures de tâches particulières. Enfin la section 6 présente une application de nos travaux à la diffusion de données pour un serveur web. La section 7 conclut le chapitre en présentant nos perspectives et des directions ouvertes.

2 Le modèle

Le modèle présenté ici généralise le modèle introduit par Edmonds en 1999 [93] en autorisant à chaque tâche à être composée d'un graphe arbitraire de sous-tâches, appelées processus.

Les tâches. On considère une suite de n *tâches* $J_1, \ldots, J_n$ arrivant à des dates $r_1, \ldots, r_n$ inconnues de l'ordonnanceur qui doit les ordonnancer sur p processeurs. Chaque tâche J_i consiste en un graphe orienté acyclique (DAG, en abrégé) $(\{J_{i,1}, \ldots, J_{i,m_i}\}, \prec)$ de processus, inconnu de l'ordonnanceur.

Les processus. Le modèle des processus est celui défini par Edmonds dans [93]. Chaque processus J_{ij} traverse un certain nombre de *phases* $J_{ij}^1, \ldots, J_{ij}^{q_{ij}}$ inconnues de l'ordonnanceur. Chaque phase J_{ij} est caractérisée par une quantité de *travail* w_{ij}^k à effectuer et un *profil d'accélération* $\Gamma_{ij}^k : \mathbb{R}_+ \to \mathbb{R}_+$, tous deux inconnus de l'ordonnanceur également. Si l'ordonnanceur attribue pendant un temps dt, ρ_{ij} processeurs ($\rho_{ij} \in \mathbb{R}_+$) au processus J_{ij} dans sa k-ème phase, le travail effectué de cette phase est :

$$dw_{ij}^k = \Gamma_{ij}^k(\rho_{ij})\, dt.$$

On fait les hypothèses naturelles suivantes sur les profils d'accélération. Chaque profil est :

- *croissant* ($\Gamma(\rho) \leqslant \Gamma(\rho')$ si $\rho \leqslant \rho'$), c.-à-d. on suppose que l'exécution d'aucun processus n'est ralentie si on lui attribue une plus grande quantité de processeurs ;

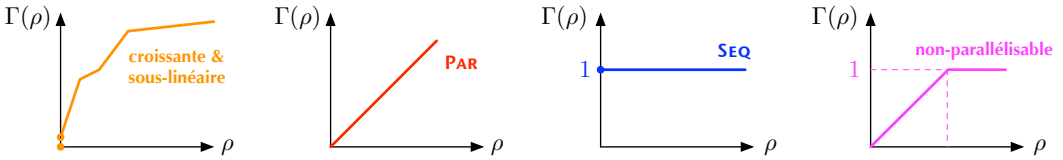


Figure 3.1 – Exemples de profils d'accélération : quelconque, **PAR**, **SEQ**, et non-parallélisable.

- *sous-linéaire* ($\frac{\Gamma(\rho)}{\rho} \geqslant \frac{\Gamma(\rho')}{\rho'}$ si $\rho \leqslant \rho'$), c.-à-d. on suppose qu'aucun processus n'utilise de manière plus efficace plus de processeurs, p. ex., on suppose qu'aucun processus ne tire trop d'avantages des caches locaux des processeurs.

La figure 3.1) présente quelques exemples de profils d'accélération :

- un profil quelconque croissant et sous-linéaire ;
- le profil d'une phase *totalement parallèle*, notée **PAR**, définie par $\Gamma(\rho) = \rho$ pour tout ρ, dont la quantité de travail effectuée progresse de $\rho\, dt$ pendant dt ; ce type de phase est une idéalisation classique du parallélisme en théorie de l'ordonnancement, v. [198].
- une phase *séquentielle*, notée **SEQ**, définie par $\Gamma(\rho) = 1$ pour tout ρ (même nul), dont la quantité de travail progresse de dt *quelque soit* le nombre de processeurs attribués par l'ordonnanceur à la tâche, même si l'ordonnanceur ne lui en attribue aucun ; ce type de phase correspond p. ex. aux périodes *idle* d'un processus ou bien au cas d'une application temps réel (comme un film ou une vidéo-conférence) qui doit avancer à vitesse constante, quelque soit la bande passante qu'on lui alloue.
- enfin une phase *non-parallélisable*, définie par $\Gamma(\rho) = \min(\rho, 1)$, qui avance proportionnellement à la part de processeur qu'on lui accorde sur un processeur, mais ne tire aucun profit de deux processeurs ou plus.

Nous verrons à la proposition 3.2 que dans l'analyse d'un ordonnanceur non clairvoyant, on peut toujours supposer que les processus ne traversent que deux types de phase : **PAR** ou **SEQ**. C'est en effet la situation la plus délicate pour un ordonnanceur non clairvoyant puisque l'une tire parti au mieux de chaque pourcentage de processeur supplémentaire accordé, alors qu'il donne à l'autre des processeurs en pure perte, et qu'il ne peut pas savoir qui est qui.

Ordonnancement. Un ordonnancement $\mathcal{S}_p$ sur p processeurs est un ensemble de fonctions à valeurs réelles positives, constantes par morceaux $\rho_{ij} : t \mapsto \rho_{ij}^t$, avec $1 \leqslant i \leqslant n$, telles que pour tout t,

$$\sum_{i,j} \rho_{ij}^t \leqslant p.$$

Chaque ρ_{ij}^t ($\in \mathbb{R}_+$) représente la quantité de processeurs allouée au processus J_{ij} au temps t. Dans ce modèle préemptif, il est possible d'allouer une fraction arbitraire de processeur à chaque processus, ce qui, en pratique est réalisé par *time multiplexing*.

Considérons un ordonnancement $\mathcal{S}_p$ des tâches $J_1, \ldots, J_n$. Chaque tâche ou processus est dit *actif* ou *en vie* depuis son arrivée ou son activation jusqu'à sa complétion. On note respectivement r_{ij} et c_{ij} les dates d'activation et de complétion du processus J_{ij}. Le processus J_{ij} est activé dès que tous les processus dont il dépend sont terminés, ainsi :

- $r_{ij} = r_i$ s'il n'a aucun prédécesseur dans le graphe de sa tâche associée J_i, c.-à-d. si $J_{i\ell} \not\prec J_{ij}$ pour tout ℓ ; et

- $r_{ij} = \max\{c_{i\ell} : J_{i\ell} \prec J_{ij}\}$ sinon.

Chaque processus J_{ij} est activé à $t = r_{ij}$ sans que l'ordonnanceur ne soit informé des dépendances activées. La $k^{\text{ème}}$ phase du processus J_{ij} termine à la date c_{ij}^k définie récursivement comme suit, avec la convention $c_{ij}^0 = r_{ij}$:

$$c_{ij}^k = \min\left\{ t : \int_{c_{ij}^{k-1}}^{t} \underbrace{\Gamma_{ij}^k(\rho_{ij}^t)\,dt}_{=dw_{ij}^k} \geqslant w_{ij}^k \right\}, \quad \text{pour } 1 \leqslant k \leqslant q_{ij}.$$

Le processus J_{ij} termine avec sa dernière phase à la date $c_{ij} = c_{ij}^{q_{ij}}$. La tâche J_i se termine donc à la date $c_i = \max_j c_{ij}$. Un ordonnancement est dit *valide* si $c_i < \infty$ pour tout i.

Nous noterons $W_{ij}(t)$ la *quantité de travail effectué* du processus J_{ij} à l'instant t, c.-à-d. $W_{ij}(t) = 0$ pour $t \leqslant r_{ij}$, $W_{ij}(t) = \sum_{k=1}^{q_{ij}} w_{ij}^k$ pour $t \geqslant c_{ij}$, et :

$$W_{ij}(t) = \sum_{k=1}^{\ell-1} w_{ij}^k + \int_{c_{ij}^{\ell-1}}^{t} \Gamma_{ij}^{\ell}(\rho_{ij}^t)\,dt, \quad \text{pour } r_{ij} \leqslant t \leqslant c_{ij},$$

où $\ell = \max\{k : c_{ij}^{k-1} \leqslant t\}$ est la phase en cours d'exécution à l'instant t.

Fonction objectif. Nous cherchons à minimiser le temps de service moyen des tâches, ou bien de façon équivalente du point de vue de l'approximation à un facteur près, le temps de service total, appelé également *temps de flot* :

$$\text{Flowtime}(\mathcal{S}_p) = \sum_{i=1}^{n} (c_i - r_i).$$

On peut voir la minimisation de cette fonction comme un mélange entre deux objectifs : la minimisation du temps de complétion maximum[41] (des processus) et du temps de flot (des tâches).

On notera n_t le *nombre de tâches en vie* à l'instant t. On obtient par simple permutation somme-intégrale, une définition alternative du temps de flot d'un ordonnancement qui sera fort utile par la suite, car elle ne dépend que de l'*état courant* des connaissances de l'ordonnanceur.

Fait 3.1 (Définition alternative). $\text{Flowtime}(\mathcal{S}_p) = \displaystyle\int_0^{\infty} n_t\,dt.$

On désigne par $\text{OPT}_p(J) = \inf_{\mathcal{S}_p} \text{Flowtime}(\mathcal{S}_p)$, le coût optimal (hors-ligne) d'un ordonnancement sur p processeurs. Notons que nous ne sommes pas assurés qu'un tel ordonnancement existe, mais seulement qu'il existe pour tout $\eta > 0$, un ordonnancement $\mathcal{O}_p$ tel que $\text{Flowtime}(\mathcal{O}_p) \leqslant \text{OPT}_p + \eta$, ce qui nous suffira pour la suite.

Non-clairvoyance, compétitivité et augmentation de ressources. Nous nous situons dans la situation réaliste d'un ordonnanceur *non clairvoyant* qui ignore *tout* de ce qu'il ordonnance. Un tel ordonnanceur découvre les tâches au moment de leur arrivée, ne connaît rien des processus qui la composent et qu'il découvre au moment de leur activation, ni rien non plus des différentes phases qu'ils traversent (rien sur leur quantité de travail, ni rien sur leur profil d'accélération) ni de leur avancement. Il est seulement informé de la terminaison

[41] *makespan*, en anglais.

de chaque processus au moment où il termine. Les *seules* informations dont il dispose sont donc les listes des tâches actives et de leurs processus actifs à chaque instant, ainsi que leurs dates d'activation.

Dans cette situation, il est évidemment difficile d'être compétitif avec l'ordonnancement hors-ligne optimal et Edmonds [93] a démontré que le temps de flot de tout algorithme non clairvoyant (même randomisé) est nécessairement à un facteur $\Omega(\sqrt{n})$ du temps de flot optimal hors-ligne en présence de tâches **SEQ** et **PAR**.

Bien qu'aucun algorithme ne puisse se comparer directement à l'optimum, il existe bien des algorithmes meilleurs que d'autres. Une approche traditionnelle en algorithmique en-ligne consiste alors à *augmenter les ressources* de l'algorithme étudié par rapport à celles de l'adversaire optimal hors-ligne auquel il est comparé. Cette méthode fut d'abord introduite avec succès par Sleator et Tarjan en 1985 [292, 51, 331] pour analyser plus finement les différentes stratégies de cache. Cette méthode fut ensuite portée à l'analyse des algorithmes d'ordonnancement en-ligne par [180, 238]. Un critère important est alors la quantité de ressources supplémentaires dont l'algorithme a besoin pour devenir compétitif avec l'optimum.

Edmonds propose donc de comparer l'ordonnancement engendré par un algorithme A sur sp processeurs à l'ordonnancement optimal sur p processeurs seulement, avec $s > 1$. On note alors $A_{sp}(J)$ l'ordonnancement engendré par A sur l'instance J avec sp processeurs. On dit alors qu'un algorithme A est *s-rapide c-compétitif* si

$$\text{Flowtime}(A_{sp}(J)) \leqslant c \cdot \text{OPT}_p(J), \quad \text{pour toute instance } J.$$

Intuitivement, A est s-rapide $O(1)$-compétitif, signifie que A *peut tenir une charge* $1/s$, au sens que le temps de service de A ne diverge pas par rapport au temps de service optimum "rempli à une hauteur $1/s$ seulement" quand n augmente ; ainsi tant que la charge reste inférieure à $1/s$, ses performances *ne se dégradent pas avec le temps*.

Comme l'ordonnanceur est autorisé à attribuer des fractions de processeurs, quitte à renormaliser les profils d'accélération d'un facteur $1/p$, on suppose sans perte de généralité à partir de maintenant que $p = 1$.

3 Réduction aux instances SEQ-PAR

Une étape fondamentale dans l'analyse d'Edmonds du cas sans dépendance [93] a été de prouver que pour obtenir la compétitivité d'un ordonnanceur non clairvoyant, il suffit de l'avoir pour les instances ne traversant que des phases **SEQ** ou **PAR**. Avec Julien Robert, nous avons clarifié, simplifié puis étendu sa preuve au cas avec dépendances. Cette nouvelle preuve lève un certain nombre d'hypothèses implicites dans la démonstration d'Edmonds, comme la monotonie de l'algorithme étudié (une propriété vérifiée par l'algorithme **EQUI** étudié dans [93]). Notre démonstration s'applique à tout algorithme non clairvoyant et nous a permis de simplifier et améliorer (à la marge) l'analyse du nouvel algorithme **LAPS**$_\beta$ (non-monotone) proposé par Edmonds et Pruhs en 2009 [99] que nous étudierons par la suite.

Proposition 3.2 (Réduction aux instances SEQ-PAR avec retard sur le PAR, [266, 268, 263]). *Pour toute instance $J = \{J_1, \ldots, J_n\}$ et tout algorithme non clairvoyant A sur s processeurs et tout ordonnancement valide $\mathcal{O}$ sur un processeur, il existe une instance $J' = \{J'_1, \ldots, J'_n\}$ dont les processus traversent uniquement des phases* **SEQ** *ou* **PAR**, *telle que :*

1. *les contraintes de dépendance sont conservées :* $(\forall i,j,k) \quad J'_{ij} \prec J'_{ik} \Leftrightarrow J_{ij} \prec J_{ik}$;
2. A_s *construit le même ordonnancement pour les processus de* J' *que pour* J :
$$A_s(J') = A_s(J)[J'/J] ;$$
3. $\mathcal{O}[J'/J]$ *est un ordonnancement valide de* J', *et donc :*
$$\text{Flowtime}(\mathcal{O}[J'/J]) \leqslant \text{Flowtime}(\mathcal{O}) ;$$
4. $A_s(J')$ *est toujours en retard par rapport à* $\mathcal{O}[J'/J]$ *sur tout travail parallèle de* J' :
$$(\forall t,i,j) \quad W_{ij}^{A(J')}(t) \leqslant W_{ij}^{\mathcal{O}[J'/J]}(t) ;$$

où $\mathcal{S}[J'/J]$ *désigne l'ordonnancement obtenu en substituant dans* $\mathcal{S}$ *chaque processus* J_{ij} *par le processus* J'_{ij} *(c.-à-d., en conservant les mêmes fonctions* ρ_{ij} *mais en ordonnançant les processus* J'_{ij} *en lieu et place des* J_{ij}*).*

Démonstration. L'originalité de notre preuve par rapport à celle de [93] est que nous allons modifier les phases des processus *en place*, c.-à-d. sans déformer l'ordonnancement, grâce aux ingrédients que nous avons développés dans [264, 266]. Ceci va nous permettre d'une part de nous abstraire des contraintes de dépendance (qui seront automatiquement satisfaites puisque l'ordonnancement est inchangé), ensuite de gagner en précision, et enfin de garantir le point 4 que nous utiliserons à la section suivante pour simplifier et améliorer l'analyse de l'algorithme **LAPS**$_\beta$ par [99] d'un facteur 2.

Comme A est non clairvoyant, nous allons modifier les phases que traverse chaque processus J_i pour assurer les quatre propriétés sans modifier sa période d'activité dans A_s. Le point 1 sera donc vérifié automatiquement puisqu'on opère uniquement sur l'intérieur des processus, sans jamais rien changer au DAG des tâches.

Considérons un processus J_{1j} de la tâche J_1 (on procède de même pour les autres tâches). La preuve procède en trois étapes. Premièrement, on redécoupe le processus en un nombre fini de segments $K_1, \ldots, K_m$ tels que chaque segment correspond à un morceau d'une phase de J_{1j} qui reçoit tout du long la même quantité de processeurs dans A_s et la même quantité de processeurs dans OPT (pour cela, il suffit de noter à quel avancement de la tâche correspond chaque changement du nombre de processeurs alloués par A_s et par OPT à l'aide des fonctions W_{1j}^A et $W_{1j}^{\mathcal{O}}$, v. fig. 3.2❶). Deuxièmement, la phase de chaque segment K_k de profil Γ, de quantité de travail w, de longueur T_A (resp. $T_{\mathcal{O}}$) et recevant ρ_A (resp. $\rho_{\mathcal{O}}$) processeurs dans A_s (resp. $\mathcal{O}$), est remplacée par (v. fig. 3.2❷) :

- une phase **SEQ** de durée $w' = T_A$, si $\rho_A \geqslant \rho_{\mathcal{O}}$;
- une phase **PAR** de travail $w' = \rho_A T_A$ sinon.

Dans le premier cas où $\rho_A \geqslant \rho_{\mathcal{O}}$, le fait que $w = \Gamma(\rho_A)T_A = \Gamma(\rho_{\mathcal{O}})T_{\mathcal{O}}$ implique que $T_{\mathcal{O}} \geqslant T_A$ par croissance de Γ. La nouvelle phase **SEQ** peut donc être terminée dans l'emplacement qui était alloué au segment K_k dans $\mathcal{O}$.

Dans le second cas où $\rho_A < \rho_{\mathcal{O}}$, le fait que $w = \Gamma(\rho_A)T_A = \Gamma(\rho_{\mathcal{O}})T_{\mathcal{O}}$ et que $\rho_A < \rho_{\mathcal{O}}$ implique par sous-linéarité de Γ que $\rho_A T_A \leqslant \Gamma(\rho_A)\frac{\rho_{\mathcal{O}}}{\Gamma(\rho_{\mathcal{O}})}T_A = \rho_{\mathcal{O}}T_{\mathcal{O}}$. La nouvelle phase **PAR** peut bien être terminée dans l'emplacement qui était alloué au segment K_k dans $\mathcal{O}$. Ainsi, nous avons le point 3.

On procède ainsi sur tous les processus de J. On obtient une instance intermédiaire J'' dont toutes les phases sont **SEQ** ou **PAR** et sont calculées pour se terminer dans A_s, non clairvoyant donc incapable de distinguer J de J'', exactement au même moment que celles de J dans A_s, d'où le point 2.

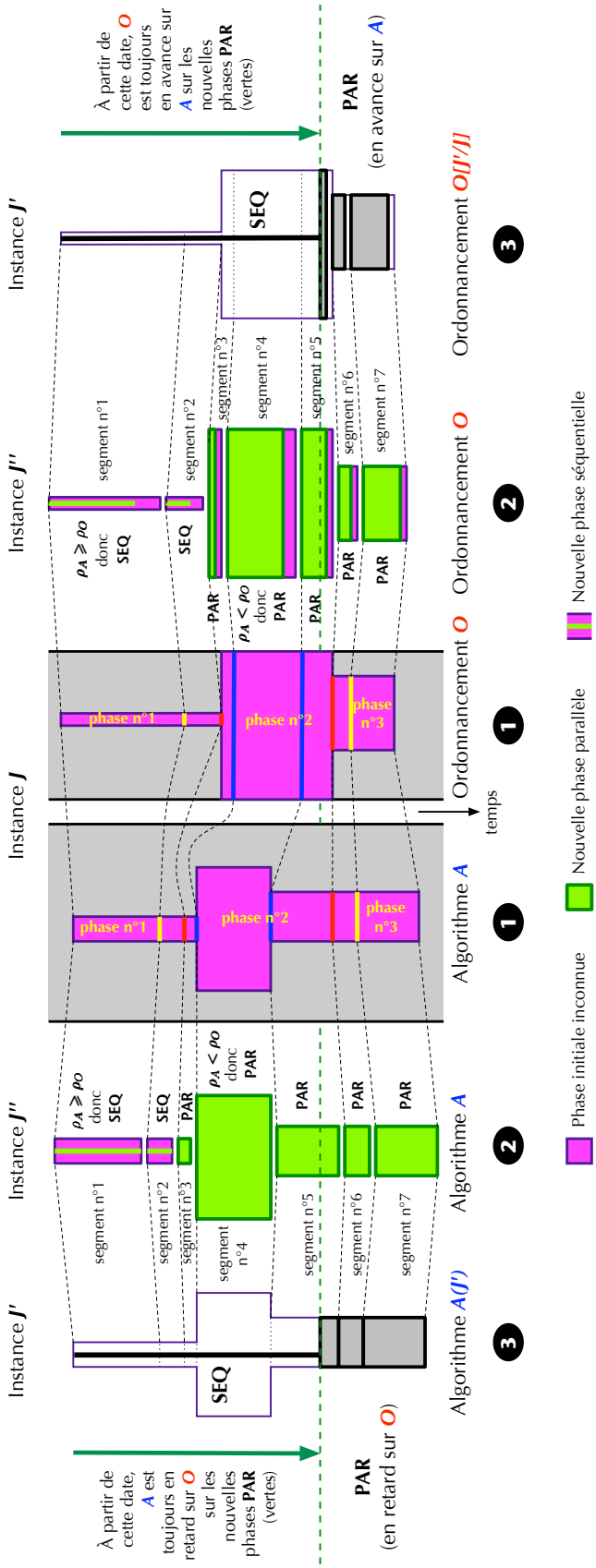


Figure 3.2 – Illustration de la réduction à une instance **SEQ-PAR** où A exécute toujours le travail **PAR** en retard sur $\mathcal{O}$ (Proposition 3.2).

Reste à garantir que A_s soit toujours en retard sur $\mathcal{O}$. Nous allons traiter chaque tâche indépendamment en procédant par induction sur son DAG de processus. Considérons les processus $\{J''_{1j}\}$ de la tâche J''_1 et modifions-en les phases par induction sur le DAG pour assurer le point 4. Supposons par induction que A_s soit en retard sur $\mathcal{O}$ sur tout travail parallèle des prédécesseurs de J''_{1j}. J''_{1j} est donc activé dans A_s après ou en même temps que dans $\mathcal{O}$: $r''^{A}_{1j} \geqslant r''^{\mathcal{O}}_{1j}$. En effet, soit J''_{1j} n'a pas de prédécesseur et il est activé en même temps dans les deux ordonnancements à l'arrivée de la tâche J_1 ; soit il a des prédécesseurs et comme A_s est en retard sur tout travail parallèle et que les travaux **Seq** avancent à la même vitesse dans les deux ordonnancements, les prédécesseurs de J''_{1j} terminent tous en retard dans A_s par rapport à $\mathcal{O}$, et il est activé après dans A_s.

Considérons à présent le *dernier* instant $\bar{t}$, $r''^{A}_{ij} \leqslant \bar{t} \leqslant c''^{A}_{1j}$, où A_s est en avance sur du travail parallèle de J''_{1j} par rapport à $\mathcal{O}$. Alors nécessairement, à cet instant : ou bien A_s n'a pas encore commencé J_{1j} alors que $\mathcal{O}$ si ; ou bien A_s a terminé J''_{1j} avant $\mathcal{O}$; ou bien A_s et $\mathcal{O}$ en sont exactement *au même point* dans le processus J''_{1j}. On remplace alors toutes les phases de J''_{1j} jusqu'à $\bar{t}$ (c.-à-d., jusqu'au point $W^{A}_{1j}(\bar{t})$) par une unique phase **Seq** de durée $\bar{t} - r''^{A}_{ij}$ (v. fig 3.2❸). Ainsi, soit A_s avait terminé J''_{1j} avant $\mathcal{O}$ et J''_{1j} consiste maintenant en une unique phase **Seq** et $\mathcal{O}$ n'a plus rien à faire et termine J''_{1j} avant A_s puisqu'il a commencé avant. Soit A_s et $\mathcal{O}$ en sont exactement au même point dans l'exécution de J''_{1j} à $t = \bar{t}$ et cette modification ne change rien ni pour l'un ni pour l'autre. En procédant ainsi par induction sur le DAG de chaque tâche, nous obtenons bien une nouvelle instance J' qui vérifie les points 1 à 4. ■

Corollaire 3.3. *Si un algorithme A est s-rapide c-compétitif pour les instances* **Seq-Par**, *alors A est s-rapide c-compétif pour des instances traversant des phases arbitraires.*

Démonstration. Soit $J_1, \ldots, J_n$ des tâches dont les processus traversent des phases de profils d'accélération quelconques, croissants et sous-linéaires. Soit $\mathcal{O}_1$ un ordonnancement quasi-optimal de $J_1, \ldots, J_n$ sur un processeur, c.-à-d. telque $\text{Flowtime}(\mathcal{O}_1) \leqslant \text{OPT}_1(J) + \eta$ pour un $\eta > 0$ arbitrairement petit. La proposition 3.2 modifie *en place*[42] les phases de tâches J_i, pour construire une instance **Seq-Par** $J'_1, \ldots, J'_n$ ayant les mêmes graphes de dépendances que $J_1, \ldots, J_n$ et telle que $A_s(J') = A_s(J)[J'/J]$ et que $\mathcal{O}_1[J'/J]$ soit un ordonnancement valide. Il s'en suit que :

$$\begin{aligned}
\text{Flowtime}(A_s(J)) &= \text{Flowtime}(A_s(J')) && \textit{(les processus ont le même temps de vie)}\\
&\leqslant c \cdot \text{OPT}_1(J') && \textit{(A est s-rapide c-compétitif pour les} \textbf{Seq-Par}\textit{)}\\
&\leqslant c \cdot \text{Flowtime}(\mathcal{O}[J'/J]) && \textit{($\mathcal{O}_1[J'/J]$ est un ordonnancement valide)}\\
&\leqslant c \cdot \text{Flowtime}(\mathcal{O})\\
&\leqslant c \cdot \text{OPT}_1(J) + c\eta, \quad \text{pour un } \eta > 0 \text{ arbitrairement petit.}
\end{aligned}$$

Ainsi, $\text{Flowtime}(A_s(J)) \leqslant c \cdot \text{OPT}_1(J)$, en prenant la limite $\eta \to 0$. ■

On ne considérera donc plus que des instances dites **Seq-Par** où les processus ne traversent que des phases **Seq** ou **Par**. Nous noterons $\textbf{Seq}(J_{ij})$ et $\textbf{Par}(J_{ij})$ les quantités de travail séquentiel et parallèle contenues dans chaque processus J_{ij} respectivement. On appelle *chaîne* d'une tâche J_i, toute suite de processus $J_{ij_1} \prec^* \cdots \prec^* J_{ij_\ell}$, où $\prec^*$ désigne la

[42] c.-à-d., à l'intérieur des ordonnancements $A_s(J)$ et $\mathcal{O}_1$ sans les modifier.

fermeture transitive de l'ordre $\prec$, et *antichaîne*, tout ensemble de processus $\{J_{1j_1}, \ldots, J_{ij_\ell}\}$ deux-à-deux indépendants (c.-à-d., deux-à-deux incomparables pour $\prec^*$). On note

$$\textsc{Seq}(J_i) = \max_{\gamma\,:\,\text{chaîne de } J_i} \sum_{J_{ij}\in\gamma} \textsc{Seq}(J_{ij}),$$

la quantité maximale de travail **Seq** dans une chaîne de J_i, et $\textsc{Par}(J_i) = \sum_j \textsc{Par}(J_{ij})$ la quantité de travail **Par** total de la tâche J_i. Comme chaque phase **Seq** avance à vitesse 1 quelque soit l'ordonnancement, $\textsc{Seq}(J_i)$ est un minorant universel du temps d'activité de chaque tâche J_i.

Fait 3.4 (Minorant). *Pour toute instance* **Seq-Par** $J_1, \ldots, J_n$, $\mathrm{OPT}_1 \geqslant \sum_{i=1}^{n} \textsc{Seq}(J_i)$.

4 Le cas sans dépendance : algorithmes LAPS$_\beta$

Le cas sans dépendance où les tâches sont réduites à un unique processus a été étudié par Jeff Edmonds en 1999 [93] puis Edmonds et Pruhs en 2009 [99]. Dans [93], Edmonds démontre que l'algorithme 3.1 **Equi** qui partage équitablement les processeurs entre les tâches en vie à chaque instant, est $(2+\varepsilon)$-rapide $O(1/\varepsilon)$-compétitif, c.-à-d. capable de supporter une charge arbitrairement proche de 50% sans diverger. Ce résultat est resté parmi les plus

Algorithme 3.1 L'ordonnanceur de tâches **Equi** sur s processeurs

pour chaque instant t **faire**
  Attribuer $\rho_i^t = s/n_t$ processeurs à chaque tâche J_i en vie à l'instant t.

impressionnants en théorie de l'ordonnancement. Il était en effet très surprenant étant donnés la généralité du modèle et les pénalités imposées à l'algorithme, qu'il soit possible de trouver un algorithme performant et plus étonnant encore qu'on arrive à le démontrer. La preuve proposée par Edmonds était d'ailleurs très compliquée et utilisait une fonction potentiel à base d'intégrales multiples difficile à décrypter.

Edmonds propose une intuition très simple des raisons des bonnes performances d'**Equi**. Une fois que l'on a réduit l'étude aux instances **Seq-Par**, on est ramené à l'étude de deux variables seulement : n_t le nombre de tâches actives et ℓ_t le nombre de ces tâches qui sont dans une phase **Seq** à l'instant t, qui mesure la quantité de processeur non-utilisée pour exécuter du travail **Par**. L'intuition est que l'ordonnancement optimal, idéalement, n'ordonnance que du travail parallèle et avance donc à vitesse 1. La quantité de travail parallèle effectuée par **Equi** sur s processeurs avance à vitesse $s(1-\ell_t/n_t)$. Ainsi si $s(1-\ell_t)/n_t < 1$, le travail parallèle s'accumule dans **Equi** par rapport à l'optimal et la proportion ℓ_t/n_t diminue. Inversement si $s(1-\ell_t/n_t) > 1$, **Equi** rattrape son retard par rapport à l'optimal sur le travail parallèle et le rapport ℓ_t/n_t augmente. En quelque sorte, **Equi** s'autorégule. Il en conclut que si **Equi** parvenait à atteindre un régime stationnaire, le travail parallèle avancerait à taux 1 et $s(1-\ell_t/n_t) \approx 1$. On pourrait alors exprimer asymptotiquement le nombre de tâches actives n_t en fonction du nombre de tâches en phase **Seq** : $n_t \approx \frac{s}{s-1}\ell_t$. D'où on déduirait que :

$$\text{Flowtime}(\textsc{Equi}_s) = \int_0^\infty n_t\,dt \approx \frac{s}{s-1}\int_0^\infty \ell_t\,dt \leqslant \frac{s}{s-1}\,\mathrm{OPT}_1,$$

puisque le temps de flot optimal est au moins égal à la somme des durées des phases séquentielles de chaque tâche (fait 3.4). Même si les constantes calculées par ces calculs approximatifs

Algorithme 3.2 L'ordonnanceur de tâches **LAPS**$_\beta$ sur s processeurs

pour chaque instant t **faire**

Numéroter les tâches en vie de 1 à n_t par dates d'arrivée croissantes.
Attribuer $\rho_i^t = s/\lceil \beta n_t \rceil$ processeurs aux tâches J_i pour $n_t - \lceil \beta n_t \rceil < i \leqslant n_t$ et rien aux autres.

sont fausses,[43] ce raisonnement donne le schéma principal de sa preuve : *borner le nombre de tâches actives n_t par le nombre ℓ_t de tâches en phase* **SEQ** dont l'intégrale est indépendante de l'algorithme.

Paradoxalement, ce résultat est resté jusqu'à très récemment confiné dans une toute petite communauté d'initiés [97, 95, 96], essentiellement du fait de la complexité de sa preuve. En simplifiant ces résultats dans [265], puis [264] et [266], nous avons permis d'élargir cette communauté et relancé l'intérêt pour ces travaux, ce qui est, je crois, une part intéressante de notre contribution.

Algorithme LAPS$_\beta$. Bien qu'il soit difficile d'imaginer un autre algorithme qu'**EQUI** vu le peu d'information dont dispose l'ordonnanceur, Edmonds conjecture dans ce même papier qu'un autre algorithme, la famille d'algorithmes 3.2, notée **LAPS**$_\beta$,[44] qui partage équitablement les processeurs entre les $\lceil \beta n_t \rceil$ tâches en vie *les plus récentes* seulement, est capable d'endurer sans diverger une charge arbitrairement proche de 100% pour β suffisamment petit. La figure 3.3 présente une exécution de **LAPS**$_{\frac{1}{2}}$ pour une instance de neuf tâches.

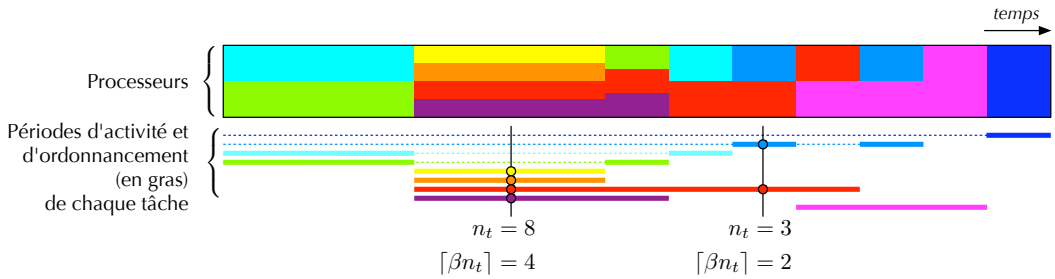


Figure 3.3 – Une exécution de l'algorithme **LAPS**$_{\beta=\frac{1}{2}}$: à chaque instant t, seules les $\lceil \frac{n_t}{2} \rceil$ tâches actives les plus récentes sont ordonnancées et se partagent équitablement les processeurs.

On pourrait reprendre les calculs approximatifs précédents pour chercher à s'en convaincre, mais il est plus intéressant de réfléchir en terme de complexité amortie pour comprendre en quoi **LAPS**$_\beta$ est plus efficace qu'**EQUI**. La première remarque est que l'on peut "facturer" le coût de l'attente de la proportion $(1 - \beta)$ de tâches non-ordonnancées sur la proportion β de tâches ordonnancées. On s'attend donc à un facteur $1/\beta$ supplémentaire sur le facteur de compétitivité. D'autre part, comme les tâches séquentielles avancent à vitesse constante, ne pas ordonnancer une certaine proportion des tâches, choisies parmi les dernières arrivées, ne peut qu'augmenter la proportion de tâches séquentielles ordonnancées simultanément et gratuitement et donc paradoxalement favoriser à terme la progression du parallèle, et ainsi réduire la quantité de processeurs nécessaires à l'auto-régulation de l'algorithme. Cet

[43] Edmonds démontre en fait qu'un régime stationnaire ne peut être atteint que si $s \geqslant 2$ en donnant un contre-exemple où le temps de flot d'**EQUI**$_s$ diverge par rapport à l'optimum pour $s < 2$.

[44] **LAPS** est l'abréviation de *Latest Arrival Processor Sharing*.

algorithme a aussi l'avantage de favoriser les petites tâches par rapport aux plus grosses et ajuste donc bien mieux le temps d'activité des tâches à leur longueur et à leur fréquence.

À la suite de nos papiers [264, 266], Edmonds et Pruhs avec lesquels nous étions en contact, se sont réintéressés à cette conjecture qu'ils ont finalement prouvée. Dans [99], ils démontrent que la famille d'algorithmes **LAPS**$_\beta$ est s-rapide $\frac{4s}{\beta\varepsilon}$-compétitive pour tout $\varepsilon > 0$ et toute vitesse $s \geqslant 1 + \beta + \varepsilon$. Cette famille *peut donc supporter une charge* $\frac{1}{1+\beta+\varepsilon}$ *arbitrairement proche de* 1 en prenant β suffisamment petit. Remarquons qu'on retrouve le résultat d'Edmonds portant sur **EQUI** $=$ **LAPS**$_1$ pour $\beta = 1$.

Leur démonstration est basée sur l'introduction d'une fonction potentiel bien plus simple que celle de [93]. Nous en présentons ici une simplification supplémentaire qui conduit à une amélioration (assez marginale) d'un facteur 2 sur le facteur de compétitivité. La clé de cette simplification est d'avoir démontré au point 4 de notre réduction (proposition 3.2) que l'on peut supposer que **LAPS**$_\beta$ est toujours en retard sur l'optimum.

Théorème 3.5 ([268, 263], amélioration marginale de [99]). *La famille d'algorithmes* **LAPS**$_\beta$ *est s-rapide $\frac{2s}{\beta\varepsilon}$-compétitive pour tout $\varepsilon > 0$ et toute vitesse $s \geqslant 1+\beta+\varepsilon$.*

Démonstration. Nous ne donnons ici que les grandes lignes pour nous familiariser avec cette démonstration, qui sera reprise en détail dans le cadre général avec contraintes de dépendance à la section 5.3. Le lecteur intéressé pourra retrouver les détails manquants dans la thèse de Julien Robert [263].

Considérons un ordonnancement $\mathcal{O}$ sur un processeur de coût $\leqslant \mathrm{OPT}_1 + \eta$ pour un $\eta > 0$ arbitrairement petit. Par la proposition 3.2, nous pouvons considérer que les tâches $J_1, \ldots, J_n$ sont toujours de type **SEQ**-**PAR** et que **LAPS**$_{\beta,s}$ est toujours en retard sur les phases parallèles par rapport à $\mathcal{O}$. Notons n_t le nombre de tâches actives dans **LAPS**$_{\beta,s}$ au temps t et renumérotons de 1 à n_t ces tâches actives par dates d'arrivée croissantes. Soit x_i^t la quantité de travail parallèle en retard de **LAPS**$_{\beta,s}$ sur $\mathcal{O}$ pour la i-ème tâche active dans cet ordre. Notons ℓ_t le nombre de tâches actives dans une phase **SEQ** dans **LAPS**$_{\beta,s}$ au temps t.

Définissons la fonction potentiel suivante qui va absorber les oscillations autour du régime stationnaire idéal $n_t \sim Cte \cdot \ell_t$ suggéré précédemment :

$$\Phi(t) = \frac{2}{\varepsilon} \sum_{i=1}^{n_t} i \cdot x_i^t.$$

Remarquons que contrairement à [99], nous n'avons pas besoin d'utiliser $\max(0, x_i^t)$ car dans notre réduction x_i^t est toujours positif, ce qui va nous permettre d'économiser *in fine* un facteur 2 sur le facteur de compétitivité. Clairement, $\Phi(0) = \Phi(\infty) = 0$. Par construction, $\Phi(t)$ est inchangée lorsqu'une nouvelle tâche arrive, et ne peut que décroître lorsqu'une tâche se termine dans **LAPS**$_{\beta,s}$ ou dans $\mathcal{O}$ (du fait de la renumérotation des tâches actives). Nous démontrons ensuite en reprenant les calculs de [99] que :

Lemme 3.6. *Si $s \geqslant 1 + \beta + \varepsilon$, pour tout t où aucune tâche n'arrive ou se termine,*

$$n_t + \frac{d\Phi(t)}{dt} \leqslant \frac{2s}{\beta\varepsilon}\ell_t.$$

On peut alors conclure l'analyse de la compétitivité de la famille **LAPS**$_\beta$:

$$\begin{aligned}
\text{Flowtime}(\textsc{Laps}_{\beta,s}) &= \int_0^\infty n_t\,dt + \Phi(\infty) - \Phi(0) \\
&= \int_{t:\text{pas arrivée ni fin}} \left(n_t + \frac{d\Phi(t)}{dt}\right) dt + \int_{t:\text{arrivée ou fin}} \underbrace{\frac{d\Phi(t)}{dt}}_{\leqslant 0}\, dt \\
&\leqslant \frac{2s}{\beta\varepsilon}\int_0^\infty \ell_t\,dt = \frac{2s}{\beta\varepsilon}\sum_{i=1}^{n} \textsc{Seq}(J_i) \leqslant \frac{2s}{\beta\varepsilon}\text{Flowtime}(\mathcal{O}) \\
&\leqslant \frac{2s}{\beta\varepsilon}(\text{OPT}_1 + \eta), \quad \text{pour un } \eta > 0 \text{ arbitrairement petit.}
\end{aligned}$$

■

5 Le cas général : algorithmes LAPS$_\beta$ ∘ EQUI

Nous étudions maintenant le cas général de tâches constituées de DAG de processus arbitraires, que nous pouvons supposer **SEQ-PAR** grâce à la proposition 3.2. Le principal obstacle qu'ont dû soulever Edmonds dans [93] et Edmonds et Pruhs [99] était la maîtrise de l'étalement du parallèle par le séquentiel en répondant à la question : *de combien augmente n_t du fait des phases séquentielles ?* Leur résultat démontre que l'impact du travail séquentiel sur l'étalement du parallèle est modéré puisque n_t est majoré par un $O(\ell_t)$ de façon amortie.

L'étalement du séquentiel. Lorsque l'on introduit des dépendances, on doit faire face à un nouveau phénomène. Des travaux séquentiels qui pourraient être ordonnancés simultanément, se retrouvent étalés sur de longues périodes de temps par le jeu des dépendances et finissent par gâcher une part considérable des processeurs. La figure 3.4 compare

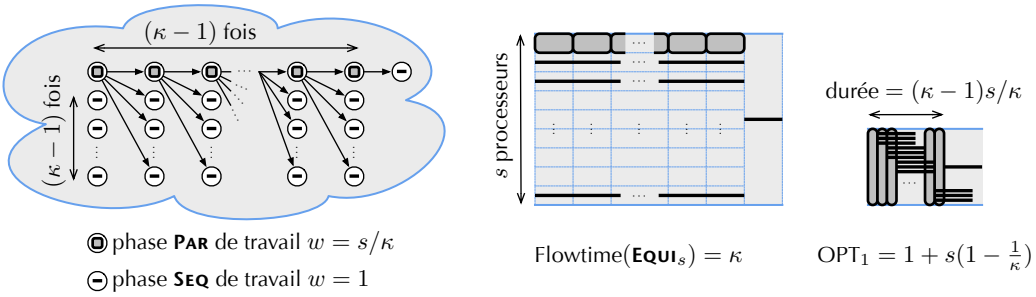


Figure 3.4 – Étalement du séquentiel dans **EQUI**$_s$ par rapport à OPT$_1$ pour une unique tâche.

l'ordonnancement calculé par **EQUI** sur s processeurs à l'optimum pour une unique tâche **SEQ-PAR** arrivant à $t = 0$ et composée de $\kappa(\kappa-1)+1$ processus interdépendants dont $(\kappa-1)$ seulement sont parallèles. Du fait des dépendances, **EQUI** gâche une proportion $1 - \frac{1}{\kappa}$ des processeurs pour exécuter des travaux séquentiels alors que l'optimum les ordonnance pendant qu'il exécute les travaux parallèles et gagne ainsi un facteur κ/s arbitrairement grand sur le temps de flot de cette tâche. Nous verrons par la suite que ceci n'est pas dû à l'algorithme **EQUI**, mais qu'on peut produire ce type d'instance pour *tout* algorithme non clairvoyant [263].

Ce phénomène pourrait se révéler incontrôlable s'il s'amplifiait en cascades d'une tâche à l'autre. Nous allons démontrer que ce n'est pas le cas et que si l'on est capable de maîtriser la part de processeurs gâchés sur le séquentiel à l'intérieur de chaque tâche, alors le temps

de flot de l'ordonnancement obtenu reste à un facteur borné de l'optimum, indépendant du nombre de requêtes.

Ne surtout pas privilégier les processus par rapport aux tâches ! Une première idée est d'utiliser l'algorithme **EQUI** ou **LAPS**$_\beta$ pour ordonnancer les processus actifs en oubliant à quelle tâche ils appartiennent. C'est l'algorithme typiquement utilisé dans un système d'exploitation classique : tous les processus actifs de même niveau de priorité reçoivent à tour de rôle une part égale des processeurs disponibles indépendamment de leur origine.

L'exemple de la figure 3.5 démontre que cette stratégie peut se révéler catastrophique : une grosse tâche peut accaparer toutes les ressources au point de faire diverger le temps de flot par rapport à l'optimum lorsque n augmente. Il est intéressant de noter que cet exemple

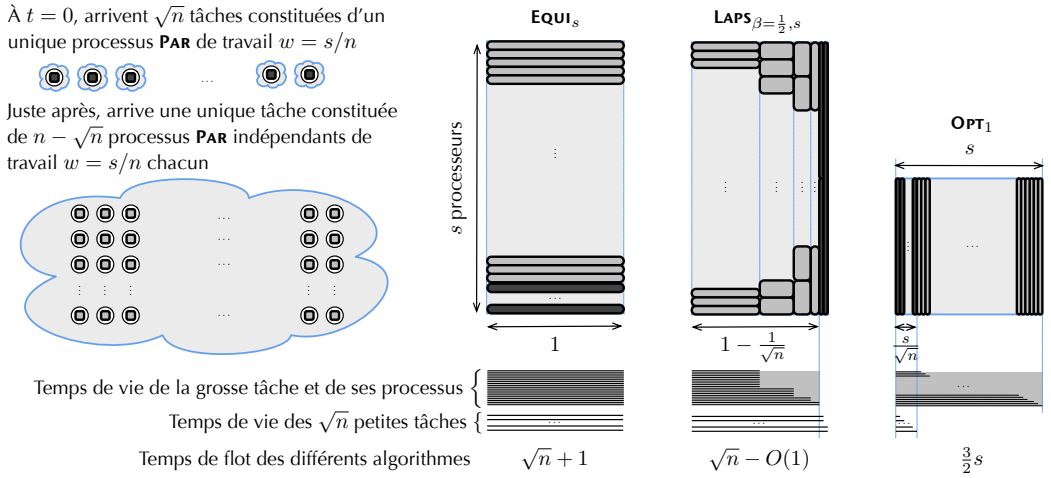


Figure 3.5 – Ordonnancer les processus actifs indépendamment de leur tâche d'origine peut conduire à des ordonnancements arbitrairement mauvais.

se généralise facilement à tout algorithme non clairvoyant qui base ses décisions uniquement sur les processus en ignorant volontairement les dépendances.

Théorème 3.7 ([265, 263]). *Aucun algorithme non clairvoyant A qui ordonnance les processus* indépendamment des tâches *auxquelles ils appartiennent ne peut être s-rapide c-compétitif pour $c < \frac{2}{3s}\sqrt{n}$ si A est déterministe, ni pour $c < \frac{1}{6s}\sqrt{n}$ si A est randomisé.*

Démonstration. Supposons A déterministe. Posons $m = \lfloor \sqrt{n} \rfloor$. Comme A est non clairvoyant et choisit d'ignorer les tâches et de baser ses décisions uniquement sur les processus, nous créons n processus à $t = 0$ et les maintenons en vie jusqu'à $t = 1$. Soient $\rho_1 \leqslant \cdots \leqslant \rho_n$ les quantités des s processeurs attribuées entre $t = 0$ et $t = 1$ par A à chaque processus par ordre croissant. Comme $\rho_1 + \cdots + \rho_n \leqslant s$, nous avons $\rho_1 + \cdots + \rho_i \leqslant \frac{is}{n}$ pour tout i.[45] Nous révélons alors que chaque processus consistait en une unique phase **PAR** de travail $\rho_1 + \frac{\varepsilon}{n}, \ldots, \rho_n + \frac{\varepsilon}{n}$, et que les m premiers processus sont des tâches indépendantes, alors que les $n - m$ processus les mieux servis par A constituent une seule et même tâche (comme l'instance de la figure 3.5). Le temps de flot de A vaut alors au moins $m + 1$ puisque

[45] Supposons sans perte de généralité que $s = 1$. Supposons par l'absurde qu'il existe un $0 \leqslant i < n$ tel que $\sigma = \rho_1 + \cdots + \rho_i \leqslant \frac{i}{n}$ et $\rho_1 + \cdots + \rho_{i+1} > \frac{i+1}{n}$. Alors $\rho_n \geqslant \cdots \geqslant \rho_{i+1} > \frac{i+1}{n} - \sigma$. Ainsi, $1 \geqslant \rho_1 + \cdots + \rho_n > \sigma + (n - i)(\frac{i+1}{n} - \sigma)$, c.-à-d. $\sigma > \frac{i}{n}$, contradiction.

toutes les tâches sont encore en vie à $t = 1$. Or, le temps de flot de l'ordonnancement sur un processeur qui commence par ordonnancer les processus des petites tâches par taille croissante puis tous les autres, vaut : $\sum_{i=1}^{m}(\rho_1 + \cdots + \rho_i + \frac{i\varepsilon}{n}) + s + \varepsilon \leqslant \sum_{i=1}^{m} \frac{i(s+\varepsilon)}{n} + s + \varepsilon$. Ainsi $\mathrm{OPT}_1 \leqslant (s+\varepsilon)(m+1)(\frac{m}{2n} + \frac{1}{m+1}) \leqslant \frac{3(s+\varepsilon)}{2\sqrt{n}} \cdot \mathrm{Flowtime}(A)$; ce qui conclut le résultat en prenant la limite $\varepsilon \to 0$.

Supposons à présent que A est randomisé. Nous utilisons le principe de Yao [328] qui affirme que l'espérance du coût du meilleur algorithme randomisé sur la pire instance est supérieure à l'espérance du coût du meilleur algorithme déterministe pour la pire distribution d'instances.[46] Il suffit donc d'exhiber une instance aléatoire pour laquelle l'espérance du coût de tout algorithme déterministe est $\geqslant C$ pour démontrer que l'espérance du coût du meilleur algorithme randomisé est $\geqslant C$. Posons $m = \lfloor\sqrt{n}\rfloor$. Considérons l'instance aléatoire suivante : n processus parallèles de travail s/n sont relâchés à $t = 0$; les dépendances sont déterminées à $t = 0$ en tirant une injection aléatoire $\varphi : \{1, \ldots, m\} \to \{1, \ldots, n\}$; les m processus numérotés $\varphi_1, \ldots, \varphi_m$ forment autant de tâches indépendantes, et les $n-m$ autres constituent une unique tâche (c'est la version aléatoire de l'instance de la fig. 3.5). Comme l'algorithme A est déterministe non clairvoyant et ordonnance les processus indépendamment des tâches auxquelles ils appartiennent, il produira le même ordonnancement sur s processeurs pour toutes ces instances. Comme chaque processus consiste en un travail **Par** $w = s/n$, au moins la moitié d'entre eux terminent après la date $t = 1/2$. Comme φ est une injection aléatoire, chacune des m petites tâches a donc une probabilité $\geqslant \frac{1}{2}$ que son processus termine après $t = \frac{1}{2}$. L'espérance du temps de flot de chaque petite tâche est donc $\geqslant \frac{1}{4}$. On en déduit que l'espérance du temps de flot de A_s est supérieure à $\frac{m}{4} + 1$. Comme il existe toujours un ordonnancement de ces tâches sur un processeur de temps de flot $\leqslant s(\frac{m(m+1)}{2n} + 1) \leqslant \frac{6s}{\sqrt{n}}(\frac{m}{4} + 1)$, on en conclut qu'il n'existe aucun algorithme randomisé ignorant volontairement l'origine des processus actifs qui soit s-rapide c-compétitif avec $c < \frac{1}{6s}\sqrt{n}$. ■

Une conséquence de ce théorème est que les algorithmes d'ordonnancement implémentés dans la plupart des systèmes d'exploitation peuvent "planter" si une tâche engendre trop de sous-processus au cours du temps, ce qui explique qu'il faille rebooter régulièrement les machines les plus sollicitées.

Nous proposons de résoudre ce problème en attribuant *d'abord* les ressources aux tâches, puis en partageant les ressources attribuées à chaque tâche entre ses processus, d'où le titre de notre article [265] :

« Soyez équitable envers les utilisateurs (les tâches), et non envers les processus ! »

Ordonnanceurs $A \circ B$. Pour cela, nous définissons un type particulier d'ordonnanceur : l'algorithme 3.3, noté $A \circ B$, obtenu par composition d'un *ordonnanceur de tâches* A et d'un *ordonnanceur de processus* B.

L'ordonnanceur de tâches A attribue une part des s processeurs à chaque tâche active, et l'ordonnanceur de processus B partage la part attribuée par A à chaque tâche active entre ses processus. Initialement, notre résultat de 2008 [266] analysait l'algorithme **Equi** $\circ$ **Equi**, mais comme notre analyse est modulaire, nous avons pu l'adapter sans effort après la parution du résultat d'Edmonds et Pruhs de 2009 [99], pour obtenir l'analyse de l'algorithme **Laps**$_\beta$ $\circ$ **Equi** présentée ici.

Le but de cette section est de démontrer le théorème suivant :

[46] Il y a en fait égalité par le théorème de von Neumann sur les jeux à somme nulle, v. p. ex. [225].

Algorithme 3.3 Ordonnanceur $A \circ B$ sur s processeurs

pour chaque instant t **faire**
  Soit $\rho_i^A(t)$ la part des s processeurs attribuée par l'ordonnanceur de tâches A à chaque tâche active J_i.
  pour chaque tâche active J_i **faire**
    Soit $\rho_{ij}^{A\circ B}(t)$ la part de $\rho_{ij}^A(t)$ processeurs attribuée par l'ordonnanceur de processus B à chaque processus actif J_{ij} de J_i.
  Attribuer $\rho_{ij}^t = \rho_{ij}^{A\circ B}(t)$ processeurs à chaque processus actif J_{ij} au temps t.

Théorème 3.8 ("[266]∘[99]", [263, 267]). *L'ordonnanceur* **LAPS**$_\beta$ ∘ **EQUI** *est s-rapide $(1+\kappa)\frac{s}{\beta\varepsilon}$-compétitif pour toute vitesse $s \geqslant 1+\beta+\varepsilon$ et tout majorant κ de la taille des antichaînes des tâches.*

On en conclut que l'algorithme **LAPS**$_\beta$ ∘ **EQUI** supporte en présence de contraintes de dépendances, exactement la *même* charge, $\frac{1}{1+\beta+\varepsilon} = 100\% - O(\beta+\varepsilon)$, que **LAPS**$_\beta$ en l'absence de dépendance ! *Les contraintes de dépendances n'ont donc pas d'impact significatif sur les performances de cet ordonnanceur.*

5.1 Linéarisation d'un ordonnancement

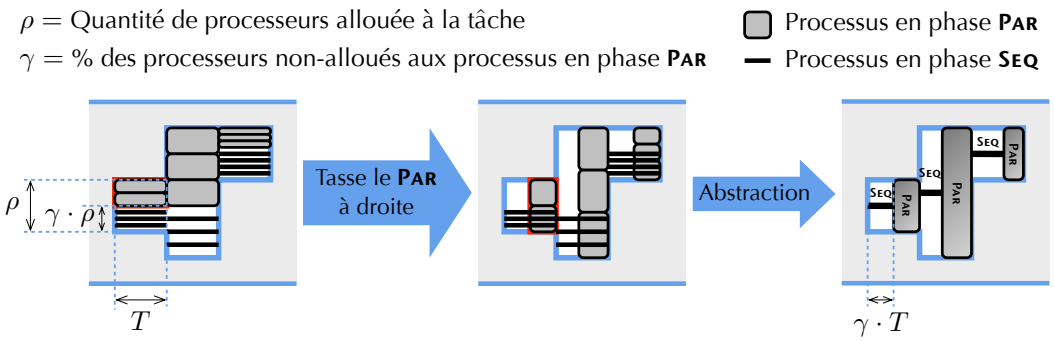
L'objet de cette section est de se ramener à deux variables seulement pour décrire un ordonnancement de tâches composées de DAG de processus **SEQ**-**PAR**. Cette réduction procède par une transformation géométrique de l'ordonancement.

Considérons un ordonnancement valide $\mathcal{S}_s$ sur s processeurs de tâches $J_1, \ldots, J_n$ dont les processus sont tous **SEQ**-**PAR**. Comme les quantités (ρ_{ij}^t) de processeurs attribuées à chaque processus sont constantes par morceaux, nous découpons cet ordonnancement en un nombre fini de segments, allant de $t = 0$ à la fin de la dernière tâche terminée, tels que durant chaque segment :

- la quantité de processeurs reçue par chaque processus actif est constante ;
- chaque processus reste dans une même phase, **SEQ** ou **PAR** ;
- aucune nouvelle tâche ou processus n'arrive ou ne se termine.

Considérons maintenant chaque tâche J_i à tour de rôle. L'ordonnancement de J_i est découpé en un certain nombre de segments à l'intérieur desquels tous ses processus actifs sont tout du long dans une phase **PAR** ou dans une phase **SEQ** et recoivent chacun une quantité constante de processeurs. La figure 3.6 illustre, à gauche, ce découpage sur une tâche découpée en trois segments.

Considérons un de ces segments et sans perte de généralité, supposons que tous les processus actifs de J_i sont ordonnancés dans un rectangle de hauteur ρ (la quantité totale de processeurs allouées aux processus de J_i) et de longueur T. Notons γ la proportion des ρ processeurs qui n'est pas allouée aux processus en phase **PAR** dans ce rectangle, et réunissons les phases **PAR** les unes au-dessus des autres dans le sous-rectangle $T \times (1-\gamma)\rho$ du haut, au dessus des phases **SEQ** placées les unes sur les autres dans le sous-rectangle $T \times \gamma\rho$ du bas, comme sur la figure 3.6 à gauche.



À gauche : une tâche (entourée en bleu) est découpée en trois segments pendant lesquels rien ne change, ni les phases traversées par les processus, ni les quantités de processeurs qui leurs sont attribuées. Durant le premier segment, de longueur T, la tâche reçoit ρ processeurs au total ; deux de ses processus actifs sont dans une phase **PAR** tandis que les trois autres sont dans une phase **SEQ** ; les phases **SEQ** occupent le sous-rectangle $T \times \gamma\rho$ du bas tandis que les **PAR** occupent le sous-rectangle $T \times (1-\gamma)\rho$ du haut.

Au milieu : on applique au premier segment de la tâche l'opération `Tasse-le-PAR-à-droite`. Dans le premier segment, les phases **PAR** dans le sous-rectangle $T \times (1-\gamma)\rho$ (en rouge) sont déplacées par une transformation affine préservant les aires dans le sous-rectangle $(1-\gamma)T \times \rho$ (en rouge) aligné avec le bord droit du segment.

À droite : on peut alors interpréter l'ordonnancement de cette tâche dans chaque segment comme une de phase **SEQ** de longueur $\gamma \cdot T$ suivie d'une phase **PAR** de travail $(1-\gamma)\rho T$ contenant tout le travail **PAR** du segment.

Figure 3.6 – Linéarisation d'un ordonnancement.

Opération `Tasse-le-PAR-à-droite`. Nous déplaçons alors les phases **PAR** contenues dans le rectangle $T \times (1-\gamma)\rho$ par l'application affine qui envoie ce rectangle sur le rectangle $(1-\gamma)T \times \rho$ aligné avec le bord droit du rectangle $\rho \times T$ (v. fig. 3.6 au milieu). Comme cette transformation conserve les aires, les quantités de travail **PAR** sont inchangées. Leur exécution est retardée juste ce qu'il faut pour qu'en leur attribuant la totalité de ρ processeurs pendant un temps $(1-\gamma)T$, elles terminent exactement à la fin du segment comme avant. Nous ne touchons pas à l'ordonnancement des phases **SEQ**.

Les ρ processeurs attribués à la tâche pendant ce segment ne sont donc maintenant gaspillés que pendant une période de longueur $\gamma \cdot T$, *égale à l'intégrale, sur la durée T du segment, de la proportion γ des ρ processeurs alloués aux processus de la tâche J_i qui ne sert pas à exécuter des phases* **PAR** *dans $\mathcal{S}_s$*. En dehors de cette période, les ρ processeurs sont utilisés à 100% pour ordonnancer du travail **PAR**.

Linéarisation. En appliquant cette transformation à tous les segments et toutes les tâches, on obtient ainsi un nouvel ordonnancement valide $\mathcal{L}(\mathcal{S}_s)$ des tâches, appelé *linéarisation de $\mathcal{S}_s$*. Remarquons que tout travail **PAR** est exécuté dans $\mathcal{L}(\mathcal{S}_s)$ en retard par rapport à $\mathcal{S}_s$ sans que cela ne change le temps de flot des processus.

Définition 3.9 (Gâchis). *Nous appelons* gâchis instantané *au temps t de l'ordonnancement $\mathcal{S}_s$ pour la tâche J_i, la proportion $\gamma_i^t \in [0,1]$ des ρ_i^t processeurs alloués par $\mathcal{S}_s$ à la tâche J_i au temps t, qui ne servent pas à exécuter des phases* **PAR** *:*

$$\gamma_i^t = 1 - \frac{\pi_i^t}{\rho_i^t}$$

où π_i^t désigne la somme de quantités ρ_{ij}^t de processeurs allouées aux processus J_{ij} dans une phase **PAR** *à l'instant t dans $\mathcal{S}_s$.*

Nous appelons gâchis *de l'ordonnancement $\mathcal{S}_s$ pour la tâche J_i, noté* gâchis$(\mathcal{S}_s, J_i)$*, l'intégrale de son gâchis instantané sur la durée de la vie de la tâche :*

$$\text{gâchis}(\mathcal{S}_s, J_i) = \int_0^\infty \gamma_i^t \, dt$$

Proposition 3.10 (Linéarisation, [266, 263]). *L'ordonnancement linéarisé $\mathcal{L}(\mathcal{S}_s)$ de $\mathcal{S}_s$ est un ordonnancement valide des tâches* **SEQ-PAR** *$J_1, \ldots, J_n$ sur s processeurs, tel que :*

- *toutes les phases, tous les processus, et toutes les tâches terminent exactement au même moment que dans $\mathcal{S}_s$; en particulier :* Flowtime$(\mathcal{L}(\mathcal{S}_s))$ = Flowtime$(\mathcal{S}_s)$ *;*
- *tout travail* **PAR** *y est exécuté en retard par rapport à $\mathcal{S}_s$;*
- *si du travail* **PAR** *d'un processus de la tâche J_i est ordonnancé à un instant t dans $\mathcal{L}(\mathcal{S}_s)$, la totalité des ρ_i^t processeurs alloués aux processus de J_i dans $\mathcal{S}_s$ est utilisée pour ordonnancer du* **PAR** *de J_i dans $\mathcal{L}(\mathcal{S}_s)$.*
- *la longueur totale des périodes de $\mathcal{L}(\mathcal{S}_s)$ où les processeurs alloués à J_i ne servent pas uniquement à ordonnancer du travail* **PAR**, *est égale au gâchis de $\mathcal{S}_s$ pour J_i,* gâchis$(\mathcal{S}_s, J_i)$.

Abstraction de $\mathcal{L}(\mathcal{S}_s)$. On peut interpréter le déroulement de chaque tâche J_i dans l'ordonnancement linéarisé $\mathcal{L}(\mathcal{S}_s)$ comme une chaîne unique alternant entre phases **SEQ** et phases **PAR** (v. illustration fig. 3.6 à droite), dont le travail **PAR** total est strictement identique à celui de J_i et où la longueur totale de ces phases **SEQ** "abstraites" est égale au gâchis de l'ordonnancement $\mathcal{S}_s$ pour la tâche J_i.

On remarquera que comme toutes les modifications sur l'ordonnancement ont été faites "en place" au sein d'un ordonnancement valide, nous avons pu jusqu'à présent ignorer les contraintes de dépendances.

5.2 α-Étaleur

Nous allons maintenant chercher à concevoir des ordonnanceurs dont on puisse maîtriser le gâchis instantané. Or, comme nous l'avons vu dans l'exemple de la figure 3.4, le jeu des dépendances peut amplifier considérablement la proportion de processeurs gâchés sur du travail **SEQ** par rapport à une stratégie optimale.

Pour les ordonnanceurs de type $A \circ B$, c'est l'ordonnanceur de processus B qui contrôle la répartition entre les processus d'une tâche, de la quantité de processeurs allouée à cette tâche par l'ordonnanceur de tâche A. On étend naturellement la définition 3.9 du gâchis au cas d'un ordonnanceur de processus.

Considérons donc un ordonnanceur de processus B qui reçoit ρ_i^t processeurs à chaque instant t via l'ordonnanceur de tâches pour ordonnancer les processus d'une tâche J_i arrivée à $t = 0$.

Définition 3.11 (Gâchis d'un ordonnanceur de processus). *De même que précédemment, nous notons γ_i^t la* proportion *des ρ_i^t processeurs qui n'est pas allouée par B à des processus de J_i traversant une phase parallèle. Nous notons* gâchis(B, ρ_i, J_i) *l'intégrale de la proportion des ρ_i^t processeurs qui n'est pas affectée à des phases* **PAR** *:*

$$\text{gâchis}(B, \rho_i, J_i) = \int_0^\infty \gamma_i^t \, dt.$$

Nous avons vu que le gâchis engendré par l'ordonnanceur de processus correspond à la quantité de travail **Seq** qui se retrouve injectée dans l'ordonnancement par la linéarisation suivant l'abstraction de la section précédente. Nous souhaitons donc que ce gâchis reste dans des proportions raisonnables, c.-à-d. que nous puissions le borner par la quantité maximale de travail séquentiel $\textbf{Seq}(J_i)$ le long d'une chaîne de J_i.

Définition 3.12 (α-Étaleur). *Un ordonnanceur de processus B est un α-étaleur si pour toute fonction $\rho : t \mapsto \rho^t$, positive et constante par morceaux, et toute tâche* **Seq-Par** *J arrivée à $t = 0$:*

$$\text{gâchis}(B, \rho, J) \leqslant \alpha \cdot \textbf{Seq}(J).$$

Nous appellerons α le coefficient d'étalement *de l'ordonnanceur de processus B.*

Le théorème suivant démontre que le coefficient d'étalement de l'ordonnanceur de processus **Equi** est borné et optimal.

Théorème 3.13 (Étaleur optimal, [266, 263]). **Equi** *est un $\frac{\kappa(J)+1}{2}$-étaleur, où $\kappa(J)$ désigne la taille maximum d'une antichaîne du DAG de J.*[47]

De plus, pour tout ordonnanceur de processus B déterministe, toute fonction positive, constante par morceaux, ρ et tout entier k, il existe une tâche J vérifiant :

$$\kappa(J) = k \quad \textit{et} \quad \text{gâchis}(B, \rho, J) \geqslant \tfrac{k+1}{2} \cdot \textbf{Seq}(J).$$

*Démonstration. Analyse d'***Equi**. Soit J_i une tâche **Seq-Par** dont la longueur maximale des anti-chaînes est k. D'après le théorème de Dilworth [84], les sommets du DAG de J_i se partitionnent en k chaînes disjointes $\xi_1, \ldots, \xi_k$. Considérons l'ordonnancement calculé par **Equi** pour la tâche J_i avec ρ_i^t processeurs. En procédant par échanges successifs le long des chaînes, nous pouvons supposer qu'il y a toujours un processus de ξ_1 actif à chaque instant. En effet, il suffit de renuméroter les chaînes pour que ξ_1 contienne le dernier processus à terminer dans cet ordonnancement. On remonte ensuite la chaîne ξ_1 jusqu'au premier instant t où elle n'a pas de processus actif. Soit J_{ij} le processus de ξ_1 qui devient actif à t et ξ_ℓ la chaîne qui contient le processus $J_{ij'}$ qui l'a activé. On échange alors le début de la chaîne ξ_ℓ jusqu'à $J_{ij'}$ inclus, avec le début de ξ_1 jusqu'à J_{ij} exclus. Puis, on continue le parcours de ξ_1 jusqu'à $t = 0$ en faisant les échanges nécessaires pour garantir que ξ_1 soit toujours active.

Comme les ξ_j sont des chaînes, au plus un processus de chaque chaîne est ordonnancé à chaque instant. Posons $\mathbb{1}_{\xi_j}(t) = 1$ si un processus de ξ_j est actif dans une phase **Seq** à l'instant t dans l'ordonnancement calculé par **Equi**, et $= 0$ sinon. Notons ν_t le nombre de processus actifs à t. Par définition,

$$\begin{aligned}
\text{gâchis}(\textbf{Equi}, \rho, J_i) &= \int_0^\infty \frac{\sum_{j=1}^k \mathbb{1}_{\xi_j}(t)}{\nu_t}\, dt \\
&= \sum_{j=1}^k \int_0^\infty \frac{\mathbb{1}_{\xi_j}(t)}{\nu_t}\, dt \\
&\leqslant \int_0^\infty \mathbb{1}_{\xi_1}(t)\, dt + \sum_{j=2}^k \int_0^\infty \frac{\mathbb{1}_{\xi_j}(t)}{2}\, dt,
\end{aligned}$$

[47] c.-à-d. le nombre maximum de processus deux-à-deux indépendants dans la tâche J, ou encore sa "largeur".

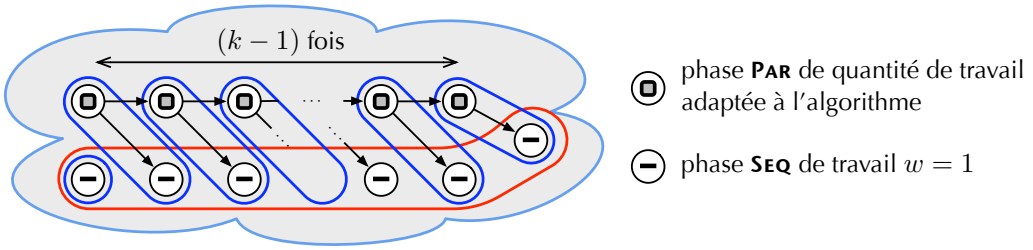


Figure 3.7 – Le peigne diabolique représenté avec une antichaîne maximum de taille k (en rouge) et une partition en k chaînes (en bleu).

car les chaînes $\xi_{j\geqslant 2}$ sont ordonnancées en parallèle avec un processus de ξ_1 au moins. Ainsi,

$$\text{gâchis}(\textsc{Equi}, \rho, J_i) \leqslant \textsc{Seq}(\xi_1) + \frac{k-1}{2}\max(\textsc{Seq}(\xi_2), \ldots, \textsc{Seq}(\xi_k)) \leqslant \frac{k+1}{2}\textsc{Seq}(J_i)$$

*Optimalité d'*EQUI. Considérons un entier k et un ordonnanceur de processus B non clairvoyant déterministe, recevant une quantité ρ^t de processeurs à chaque instant t. Considérons une tâche J ayant pour DAG un peigne comme celui de la figure 3.7 d'antichaîne maximum de longueur k. Comme B est non clairvoyant, nous annoncerons au fur et à mesure de son exécution quel processus est dans quelle phase. À $t = 0$, deux processus sont relâchés. À $t = 1$, on annonce que le processus qui maximise le gâchis (en y incluant les processeurs éventuellement inutilisés), était en phase SEQ et se termine à $t = 1$, tandis que l'autre était en phase PAR, et engendre maintenant deux nouveaux processus. On répète l'opération jusqu'à $t = k - 1$ où le processus PAR engendre un unique processus qu'on annonce SEQ de durée 1. Le gâchis de B est alors d'au moins 50% des processeurs entre $t = 0$ et $t = k - 1$ et de 100% entre $k - 1$ et k, soit $\geqslant \frac{k+1}{2}\textsc{Seq}(J)$, puisque $\textsc{Seq}(J) = 1$. ■

Remarquons que nous ne disposons pas actuellement de minorant du coefficient d'étalement des ordonnanceurs de processus non clairvoyants randomisés.

5.3 Compétitivité de LAPS$_\beta$ ∘ α-étaleur

Nous pouvons maintenant démontrer le résultat principal de cette section : le théorème 3.8. Le principe est maintenant relativement simple : l'ordonnancement linéarisé va nous permettre de nous ramener à l'étude de deux variables seulement comme dans le cas sans contrainte de dépendance étudié à la section 4 [99] ; le fait qu'EQUI soit un α-étaleur, pour $\alpha = \frac{\kappa+1}{2}$, va nous permettre quant-à-lui de borner la quantité de SEQ créée par la linéarisation.

Réduction à deux variables. Considérons un ordonnanceur de processus α-étaleur B et étudions l'algorithme $\textsc{Laps}_\beta \circ B$. Prenons une instance quelconque $J_1, \ldots, J_n$. Soient $\mathcal{S}_s = \textsc{Laps}_{\beta,s} \circ B(J)$ l'ordonnancement calculé sur cette instance par $\textsc{Laps}_\beta \circ B$ sur s processeurs, et $\mathcal{O}_1$ un ordonnancement quasi-optimal sur un processeur pour cette instance, c.-à-d. de temps de flot au plus $\text{OPT}_1(J) + \eta$ pour un $\eta > 0$ arbitrement petit. Soit $J'_1, \ldots, J'_n$ l'instance SEQ-PAR construite par la proposition 3.2 telle que l'ordonnancement $\mathcal{S}'_s = \textsc{Laps}_{\beta,s} \circ B(J')$ calculé par $\textsc{Laps}_{\beta,s} \circ B$ pour J' soit le même que pour J, c.-à-d. $\mathcal{S}'_s = \mathcal{S}_s[J'/J]$, et tel que $\mathcal{O}'_1 = \mathcal{O}_1[J'/J]$ soit un ordonnancement valide de J', et tel que tout travail PAR soit exécuté dans $\mathcal{S}'_s$ en retard par rapport à $\mathcal{O}'_1$. Considérons $\mathcal{L}(\mathcal{S}'_s)$

l'ordonnancement linéarisé de $\mathcal{S}'_s$ suivant la proposition 3.10. Nous définissons alors deux variables :

- n_t, le nombre de tâches actives à l'instant t dans $\mathcal{L}(\mathcal{S}'_s)$;
- ℓ_t, le nombre de tâches actives à l'instant t dans $\mathcal{L}(\mathcal{S}'_s)$ dont les processus actifs ne sont pas tous dans une phase **PAR**.

Comme $\mathcal{S}_s$ et $\mathcal{S}'_s$ sont identiques, ils ont le même temps de flot et d'après la proposition 3.10 et le fait 3.1,

$$\text{Flowtime}(\textsc{Laps}_{\beta,s} \circ B(J)) = \text{Flowtime}(\mathcal{S}'_s) = \text{Flowtime}(\mathcal{L}(\mathcal{S}'_s)) = \int_0^\infty n_t \, dt.$$

Maîtriser $\int_0^\infty \ell_t \, dt$. Posons $\mathbb{1}_{\textsc{Seq}_i}(t) = 1$ si les processus actifs de J'_i dans $\mathcal{L}(\mathcal{S}'_s)$ ne sont pas tous dans une phase **PAR** à l'instant t, et $= 0$ sinon. Soit $\rho_i(t)$ la quantité de processeurs attribuée à la tâche J'_i par $\textsc{Laps}_{\beta,s}$ dans $\mathcal{S}'_s$ à l'instant $t + r_i$. Nous avons :

$$\begin{aligned}
\int_0^\infty \ell_t \, dt &= \sum_{i=1}^n \int_0^\infty \mathbb{1}_{\textsc{Seq}_i}(t) \, dt && \text{(par définition)} \\
&= \sum_{i=1}^n \text{gâchis}(B, \rho_i, J'_i) && \text{(proposition 3.10)} \\
&\leqslant \sum_{i=1}^n \alpha \cdot \textsc{Seq}(J'_i) && (B \text{ est un } \alpha\text{-étaleur}) \\
&\leqslant \alpha \cdot \textsc{Seq}(J')
\end{aligned}$$

où $\textsc{Seq}(J') = \sum_{i=1}^n \textsc{Seq}(J'_i)$. Or, $\mathcal{O}'_1$ étant un ordonnancement valide de $J'_1, \ldots, J'_n$, son temps de flot est nécessairement supérieur à $\textsc{Seq}(J')$ (fait 3.4). Ainsi,

$$\begin{aligned}
\int_0^\infty \ell_t \, dt &\leqslant \alpha \cdot \text{Flowtime}(\mathcal{O}'_1) \\
&\leqslant \alpha \cdot \text{Flowtime}(\mathcal{O}_1) && (\text{car } \mathcal{O}'_1 = \mathcal{O}_1[J'/J]) \\
&\leqslant \alpha \cdot (\text{OPT}_1(J) + \eta), && \text{pour un } \eta > 0 \text{ arbitrairement petit.}
\end{aligned}$$

Relier n_t et ℓ_t. La preuve suit maintenant les lignes de [99] avec la simplification que nous avons vue à la section 4. On renumérote à chaque instant les tâches actives dans $\mathcal{L}(\mathcal{S}'_s)$ de 1 à n_t par date d'arrivée croissante. Nous notons x_i^t la quantité de travail parallèle (tous sous-processus confondus) en retard dans $\mathcal{L}(\mathcal{S}'_s)$ par rapport à $\mathcal{O}'_1$ pour la $i^{\text{ème}}$ tâche active dans cet ordre. Nous définissons la fonction potentiel :

$$\Phi(t) = \frac{2}{\varepsilon} \sum_{i=1}^{n_t} i \cdot x_i^t.$$

Clairement, $\Phi(0) = \Phi(\infty) = 0$. Et par construction, $\Phi(t)$ est inchangée lorsqu'une nouvelle tâche arrive, et ne peut que décroître lorsqu'une tâche se termine dans $\mathcal{L}(\mathcal{S}'_s)$ ou dans $\mathcal{O}'_1$ (du fait de la renumérotation des tâches).

Lemme 3.14 (Adaptation directe de [99]). *Pour toute date t qui n'est pas une date d'arrivée ou de complétion d'une tâche dans $\mathcal{L}(\mathcal{S}'_s)$ ou $\mathcal{O}'_1$,*

$$n_t + \frac{d\Phi(t)}{dt} \leqslant \frac{2s}{\beta\varepsilon} \cdot \ell_t.$$

Démonstration. Étudions les contributions respectives de $\mathcal{O}'_1$ et $\mathcal{L}(\mathcal{S}'_s)$ à $\frac{d\Phi(t)}{dt}$. La contribution de $\mathcal{O}'_1$ à $\frac{d\Phi(t)}{dt}$ est inférieure à :

$$\frac{2}{\varepsilon}\sum_{i=1}^{n_t} i\rho_i^O \leqslant \frac{2}{\varepsilon}n_t,$$

car $\sum_{i=1}^{n_t}\rho_i^O \leqslant 1$ ($\mathcal{O}'_1$ dispose d'un unique processeur), où ρ_i^O désigne la quantité de processeurs allouée à l'instant t par $\mathcal{O}'_1$ à la $i^{\text{ème}}$ tâche dans l'ordre de Φ.

Étudions à présent la contribution de $\mathcal{L}(\mathcal{S}'_s)$ à $\frac{d\Phi(t)}{dt}$. Supposons tout d'abord que $\ell_t \leqslant \lceil\beta n_t\rceil$. $\mathcal{L}(\mathcal{S}'_s)$ ordonnance alors au moins $\lceil\beta n_t\rceil - \ell_t$ tâches dont tous les processus sont dans des phases **PAR**, et dont les indices sont au pire $n_t - \lceil\beta n_t\rceil + 1, \ldots, n_t - \ell_t$. La contribution (négative) de $\mathcal{L}(\mathcal{S}'_s)$ à $\frac{d\Phi(t)}{dt}$ est alors inférieure à :

$$\begin{aligned}
&\leqslant \frac{2}{\varepsilon}\sum_{i=n_t-\lceil\beta n_t\rceil+1}^{n_t-\ell_t} i\cdot\left(-\frac{s}{\lceil\beta n_t\rceil}\right)\\
&= \frac{s}{\varepsilon\lceil\beta n_t\rceil}(2n_t\ell_t - 2\lceil\beta n_t\rceil n_t - \ell_t^2 + \ell_t + \lceil\beta n_t\rceil^2 - \lceil\beta n_t\rceil)\\
&\leqslant \frac{s}{\varepsilon\lceil\beta n_t\rceil}(2n_t\ell_t - 2\lceil\beta n_t\rceil n_t + \lceil\beta n_t\rceil^2 - \lceil\beta n_t\rceil)\\
&= \underbrace{\frac{2sn_t\ell_t}{\varepsilon\lceil\beta n_t\rceil}}_{\leqslant\frac{2s}{\beta\varepsilon}\ell_t} - \frac{2s}{\varepsilon}n_t + \underbrace{\frac{s\lceil\beta n_t\rceil}{\varepsilon} - \frac{s}{\varepsilon}}_{\leqslant\frac{s\beta n_t}{\varepsilon}} \leqslant \frac{2s}{\beta\varepsilon}\ell_t + \left(\frac{s\beta}{\varepsilon} - \frac{2s}{\varepsilon}\right)n_t
\end{aligned}$$

En sommant les deux contributions de $\mathcal{O}'_1$ et $\mathcal{L}(\mathcal{S}'_s)$, nous obtenons que pour $s \geqslant 1+\beta+\varepsilon$:

$$n_t + \frac{d\Phi(t)}{dt} \leqslant \underbrace{\left(1 + \frac{s\beta}{\varepsilon} - \frac{2s}{\varepsilon}\right)}_{\leqslant 0} n_t + \frac{2s}{\beta\varepsilon}\ell_t \leqslant \frac{2s}{\beta\varepsilon}\cdot\ell_t.$$

Supposons à présent que $\ell_t > \lceil\beta n_t\rceil$. Dans le pire cas, toutes les tâches ordonnancées par $\mathcal{L}(\mathcal{S}'_s)$ sont **SEQ**, et la contribution de $\mathcal{L}(\mathcal{S}'_s)$ à $\frac{d\Phi(t)}{dt}$ est nulle. Alors,

$$n_t + \frac{d\Phi(t)}{dt} \leqslant n_t + \frac{2}{\varepsilon}n_t + 0 \leqslant \frac{2(1+\beta+\varepsilon)}{\varepsilon}\cdot\frac{\lceil\beta n_t\rceil}{\beta} \leqslant \frac{2s}{\beta\varepsilon}\cdot\ell_t \quad \blacksquare$$

Théorème 3.15 ([266, 263]). *Pout tout ordonnanceur de processus α-étaleur B, l'algorithme* **LAPS**$_\beta \circ B$ *est s-rapide $\frac{2s\alpha}{\beta\varepsilon}$-compétitif pour tout $s \geqslant 1+\beta+\varepsilon$.*

Démonstration. Il suffit donc de combiner la majoration de n_t par ℓ_t modulée par les variations de la fonction potentiel Φ (lemme 3.14), avec la borne que nous avons obtenue sur

l'intégrale de ℓ_t grâce au coefficient d'étalement α de B :

$$\begin{aligned}\text{Flowtime}(\textsc{Laps}_{\beta,s} \circ B(J)) &= \int_0^\infty n_t\,dt + \Phi(\infty) - \Phi(0)\\ &= \int_{t:\text{pas arrivée ni fin}} \left(n_t + \frac{d\Phi(t)}{dt}\right) dt + \int_{t:\text{arrivée ou fin}} \underbrace{\frac{d\Phi(t)}{dt}}_{\leqslant 0}\, dt \leqslant \frac{2s}{\beta\varepsilon}\int_0^\infty \ell_t\,dt\\ &\leqslant \frac{2s\alpha}{\beta\varepsilon}(\text{OPT}_1 + \eta), \quad \text{pour tout } \eta > 0. \end{aligned}$$

■

Le théorème 3.8 est la conséquence directe des théorèmes 3.15 et 3.13.

5.4 Facteur d'étalement des structures classiques de tâches

Le théorème 3.15 permet d'obtenir des algorithmes plus performants pour ordonnancer des structures de tâches particulières si l'on dispose d'ordonnanceurs adaptés à ces structures.

5.4.1 Chaînes indépendantes

On considère le cas où chaque tâche consiste en un ensemble de processus disjoints, c.-à-d. en au plus κ chaînes indépendantes.

Théorème 3.16. **Equi** *est un H_κ-étaleur pour toute tâche composée d'au plus κ chaînes indépendantes.*[48]

De plus, aucun ordonnanceur de processus déterministe non clairvoyant n'a de meilleur coefficient d'étalement pour ce type de tâche.

Démonstration. Comme les κ chaînes sont relâchées ensemble à $t = 0$, la $i^{\text{ème}}$ chaîne à terminer est ordonnancée simultanément avec au moins $\kappa - i$ autres chaînes et sa contribution au gâchis est donc inférieure à $\frac{1}{\kappa-i+1}$ fois sa quantité de **Seq**. Le gâchis total est donc inférieure à H_κ fois la longueur maximale de **Seq** d'une chaîne.

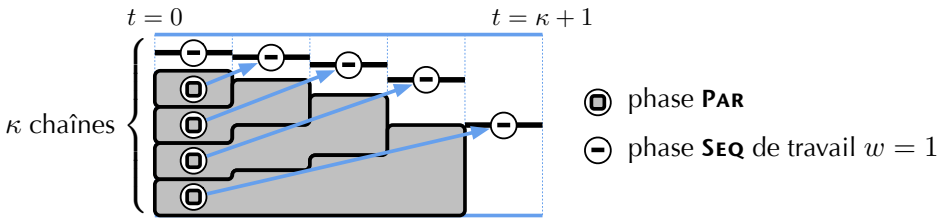


Figure 3.8 – κ chaînes maximisant le coefficient d'étalement de tout ordonnanceur de processus non clairvoyant.

Considérons à présent un ordonnanceur de processus déterministe A non clairvoyant. Considérons l'instance J suivante illustrée par la figure 3.8. On relâche κ processus à $t = 0$. À $t = 1$, on déclare que celui qui a reçu le plus de processeurs termine et était en phase **Seq** de durée 1, tandis que les autres étaient en phase **Par** de travail égal à la quantité de processeurs

[48] où $H_\kappa = 1 + \frac{1}{2} + \cdots + \frac{1}{\kappa} \sim \log\kappa$ désigne la somme harmonique.

reçue. On poursuit jusqu'à $t = \kappa$ en déclarant terminé et en phase **SEQ** le processus qui a été le mieux servi entre $t-1$ et t, tandis que les autres restent en vie et étaient en phase **PAR**. À $t = \kappa$, il ne reste alors plus qu'un seul processus en phase **SEQ** de durée 1. Le gâchis total est alors $\geqslant H_\kappa \cdot 1 = H_\kappa \cdot \textsc{Seq}(J)$. Le coefficient d'étalement de A est donc $\geqslant H_\kappa$. ■

Corollaire 3.17. *Pour tout $s \geqslant 1+\beta+\varepsilon$, **LAPS**$_\beta$ ∘ **EQUI** est s-rapide $\frac{2sH_\kappa}{\beta\varepsilon}$-compétitif lorsque les tâches sont des ensembles d'au plus κ chaînes indépendantes.*

5.4.2 Coefficient d'étalement et facteur de compétitivité

On pourrait s'attendre à ce que la connaissance du coefficient d'étalement optimal pour une structure donnée, nous permette d'obtenir directement des minorants du facteur de compétitivité d'un algorithme arbitraire en ne considérant qu'une unique tâche, arrivant à $t = 0$, structurée comme il faut pour maximiser le coefficient d'étalement. Or, il se trouve que l'instance proposée par le théorème ci-dessus (fig. 3.8) ne le permet pas car elle engendre *beaucoup trop de travail* **PAR** : $(\kappa + 1 - H_\kappa)$ dans le pire cas. Ainsi, le meilleur minorant du facteur compétitivité c qu'on puisse en tirer n'a aucun intérêt : $c \geqslant \frac{1+k}{\kappa+2-H_\kappa} \sim 1$.

En fait, nous avons démontré dans [264, 263] que le facteur de compétitivité optimal pour une tâche composées de $\leqslant \kappa$ chaînes indépendantes arrivant à $t = 0$ vaut $\frac{\log\kappa}{\log\log\kappa} \ll H_\kappa$ (théorème 3.18 ci-dessous). Nous en concluons que l'*on ne peut pas se restreindre au cas où les tâches arrivent toutes à $t = 0$ pour "convertir" un minorant du coefficient d'étalement en un minorant sur le facteur de compétitivité*. Julien Robert a proposé dans sa thèse [263] une technique particulière (les *tâches remplisseuses*) pour y parvenir dans certain cas, que nous évoquerons en conclusion de cette section.

Cas de l'ordonnancement par lots. Dans [264], nous avons étudié le cas de n tâches constituées d'au plus κ chaînes indépendantes, arrivant toutes à $t = 0$. On parle alors d'*ordonnancement par lots* (*batch scheduling*, en anglais), puisque les n tâches arrivent par lots, toutes ensemble à $t = 0$. La notion de charge maximale supportée par un algorithme et sa notion duale, l'augmentation de ressources, n'existent pas pour l'ordonnancement par lots car disposer de s fois plus de processeurs que OPT ne permet de gagner qu'un facteur $\leqslant 1/s$, indépendant de n, sur le facteur de compétitivité et ne change donc en rien à sa dépendance en le nombre n de requêtes. En particulier, **LAPS**$_\beta$ est moins performant dans cette situation qu'**EQUI**.

Nous démontrons que si toutes les tâches arrivent à $t = 0$, on peut obtenir un gain d'un facteur $\log\log\kappa$ sur le facteur de compétitivité.

Théorème 3.18 (Ordonnancement par lot d'ensembles de chaînes indépendantes, [264, 263]). *Dans le cas de n tâches composées d'au plus κ chaînes indépendantes arrivant toutes à $t = 0$, **EQUI** ∘ **EQUI** est $\frac{(2+\sqrt{3}+o_\kappa(1))\log\kappa}{\log\log\kappa}$-compétitif.*

De plus, aucun algorithme A non clairvoyant n'est c-compétitif pour $c < \frac{\log\kappa}{2\log\log\kappa}$ si A est déterministe, et pour $c < \frac{\log\kappa}{8\log\log\kappa}$ si A est randomisé.

Démonstration. L'analyse d'**EQUI** ∘ **EQUI** repose dans les grandes lignes sur le schéma suivant. Nous renvoyons le lecteur à [264, 263] pour la preuve complète.

Comme le nombre de tâches actives et de chaînes de processus actifs est une fonction décroissante du temps (aucune tâche ni chaîne de processus n'arrivent après $t = 0$), les pires instances pour **EQUI** sont du type **PAR**-puis-**SEQ**, puisque le **PAR** reçoit alors moins de processus

par unité de temps. On découpe le déroulement de chaque chaîne J_i en intervalles suivant que la proportion de chaînes actives en phase **Seq** est supérieure (intervalle de type I) ou inférieure (type II) à un certain seuil $1-\alpha$ pour $\alpha = \frac{(\log\log\kappa)^2}{\log\kappa}$. Alors,

- Comme les chaînes en phase **Seq** de J_i terminent au bout d'un temps fixé $\leqslant$ **Seq**(J_i), le nombre de chaînes actives de J_i diminue d'un facteur $(1-\alpha)$ au moins au bout d'un temps $\leqslant$ **Seq**(J_i) après le début de tout intervalle de type I. Le retard total imposé par le **Seq** est donc borné par $\frac{\log\kappa}{\log 1/\alpha}$**Seq**$(J_i) = \frac{(1+o_\kappa(1))\log\kappa}{\log\log\kappa}$**Seq**$(J_i)$.
- La quantité de travail **Par** effectuée dans la tâche avance au moins de ρ_i/α dans chaque intervalle de type II. Nous pouvons donc remplacer chaque intervalle de type II par une phase d'au plus ρ_i/α travail **Par**, concentrant toutes les chaînes en une unique chaîne dont le travail **Par** a augmenté d'un facteur au plus $1/\alpha$.

Le résultat d'Edmonds [95] portant sur l'analyse de tâches consistant en une unique chaînes de phases, nous permet alors de conclure que le temps de flot de **Equi** $\circ$ **Equi** est $\leqslant (2+\sqrt{3})(\frac{1}{\alpha}+\frac{\log\kappa}{\log 1/\alpha})\max(\textbf{Par}(J),\textbf{Seq}(J)) \leqslant \frac{(2+\sqrt{3}+o_\kappa(1))\log\kappa}{\log\log\kappa}\cdot \mathrm{OPT}$. Nous renvoyons le lecteur à [264, 263] pour les détails de la preuve.

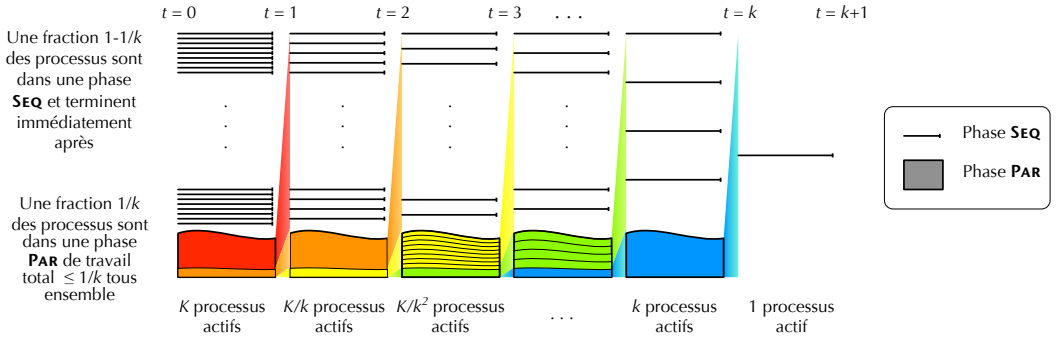


Figure 3.9 – Exemple d'une tâche composées de $K = k^k$ chaînes diaboliques.

Optimalité de **Equi** $\circ$ **Equi**. Notre minorant du facteur de compétitivité de tout algorithme non clairvoyant repose sur l'instance suivante illustrée par la figure 3.9. Considérons tout d'abord un ordonnanceur A déterministe non clairvoyant et un entier k. À $t=0$, on relâche $K=k^k$ processus. À $t=1$, on révèle que la fraction $1-\frac{1}{k}$ des K processus les mieux servis par A termine et que ceux-ci étaient tous dans une phase **Seq** de durée 1, tandis que la fraction $1/k$ des processus les moins bien servis reste en vie et qu'ils étaient tous dans une phase **Par** de travail égal à la quantité de processeurs reçue (de travail total tous ensemble $\leqslant \frac{1}{k}$). On poursuit ainsi en éliminant à chaque $t=i$, pour $i=1$ à $k-1$, la fraction $1-\frac{1}{k}$ des processus les mieux servis en leur attribuant une phase **Seq** de durée 1 et en attribuant aux autres une phase **Par** de travail égal à la quantité de processeurs reçue (de travail total toujours $\leqslant \frac{1}{k}$) ; jusqu'à $t=k$ où il ne reste plus qu'un unique processus en phase **Seq** de durée 1. Le temps de flot de A pour cette instance est donc $k+1 \sim \frac{\log K}{\log\log K}$ alors que $\mathrm{OPT} \leqslant 2$ en ordonnançant toutes les phases **Par** d'abord en une unité de temps (le travail total est $\leqslant 1$), puis les phases **Seq** de durée 1. Nous en déduisons que A ne peut être c-compétitif pour $c < \frac{\log K}{2\log\log K}$.

En utilisant le principe de Yao [328] et en randomisant l'instance des k^k chaînes diaboliques ci-dessus de la même manière qu'au théorème 3.7 page 161, nous obtenons l'autre borne pour tout algorithme randomisé et non clairvoyant (v. [264, 263] pour les détails). ■

5.4.3 Arborescences entrantes

Considérons à présent le cas de tâches dont les structures sont des arborescences entrantes d'au plus κ feuilles. Regardons l'ordonnancement calculé par un algorithme A non clairvoyant et ne conservons qu'une unique contrainte de dépendance par processus non-initial : le lien provenant du processus qui l'a activé dans A. On obtient alors une instance où chaque tâche consiste en $\leqslant \kappa$ chaînes indépendantes, plus facile pour OPT et produisant le même ordonnancement pour A. On en déduit que le résultat du théorème 3.18 se porte directement au cas des forêts entrantes.

Théorème 3.19 (Arborescences entrantes). **EQUI** *est un H_κ-étaleur pour toute tâche consistant en une arborescence entrante de processus à au plus κ feuilles.*

Corollaire 3.20. *Pour tout $s \geqslant 1+\beta+\varepsilon$,* **LAPS**$_\beta$ ∘ **EQUI** *est s-rapide $\frac{2sH_\kappa}{\beta\varepsilon}$-compétitif lorsque les tâches sont des arborescences entrantes d'au plus κ feuilles.*

5.4.4 Autres structures

Arborescences sortantes. Comme nous l'avons vu précédemment, pour l'analyse d'un algorithme non clairvoyant, nous pouvons toujours supposer que chaque processus a au plus un prédécesseur dans la structure de chaque tâche. Les arborescences sortantes sont donc équivalentes au cas de DAG arbitraires et **EQUI** a pour ces structures le coefficient d'étalement optimal $\frac{\kappa(J)+1}{2}$.

Nous pouvons cependant espérer que le coefficient d'étalement soit significativement plus faible pour des arborescences sortantes particulières. Nous avons donc proposé dans [266, 263] un algorithme pour calculer le coefficient d'étalement exact d'**EQUI** pour une arborescence sortante donnée. Cet algorithme à base de programmation dynamique, est présenté en détail dans la thèse de Julien Robert [263]. Le principe est de démontrer que le coefficient d'étalement est maximisé lorsque les processus terminent dans un ordre issu d'un parcours préfixe de l'arborescence, c.-à-d. si chaque processus s'exécute simultanément avec les processus de ses sous-arbres frères de numéro supérieur pour une numérotation des fils de chaque nœud (v. l'exemple fig.3.10). Puis, nous démontrons que le coefficient est maximisé lorsque les phases **SEQ** se trouvent toutes, tout au bout des feuilles de l'arborescence, et ont toutes la même longueur qu'on fixe à 1. On peut alors par un parcours de l'arborescence, calculer récursivement la numérotation des fils de chaque sommet qui maximise le gâchis par programmation dynamique. La figure 3.10 illustre une instance qui maximise le coefficient d'étalement d'**EQUI** pour une arborescence donnée.

Proposition 3.21 ([266, 263]). *Il existe un programme dynamique (exponentiel en la taille de l'arbre) qui calcule le coefficient d'étalement maximum exact d'***EQUI** *pour une arborescence sortante donnée.*

Arborescences sortantes complètes d-régulières. On peut résoudre à l'aide de `Maple` le programme dynamique dans le cas des arborescences sortantes d-régulières. Nous obtenons alors qu'**EQUI** est légèrement plus performant que dans le cas d'une arborescence sortante arbitraire.

Théorème 3.22 ([266, 263]). **EQUI** *est un $\Theta(\kappa \cdot \frac{\log\log\kappa}{\log\kappa})$-étaleur pour les arborescences sortantes complètes d-régulières à au plus κ feuilles.*

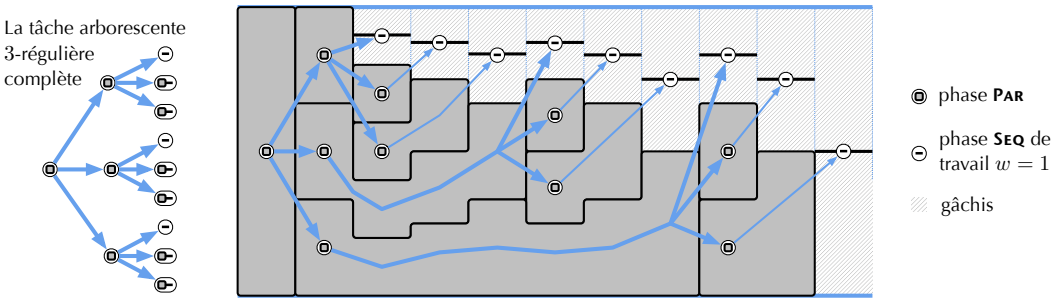


Figure 3.10 – Instance maximisant le coefficient d'étalement d'**EQUI** pour une arborescence sortante 3-régulière.

Si on autorise l'algorithme à connaître le processus père qui active le processus fils, alors on peut proposer un ordonnanceur de processus avec un meilleur coefficient d'étalement : l'algorithme **SPLIT** qui partage équitablement les ressources entre nœuds de même niveau dans l'arbre récursivement.

Théorème 3.23 ([266, 263]). **SPLIT** *est un* $\kappa^{\log\log d/\log d}$*-étaleur pour les arbres sortants complets* d*-réguliers à* $\leqslant \kappa$ *feuilles.*

5.5 Discussion

Nous avons donc défini un nouveau paramètre, le coefficient d'étalement, qui permet d'obtenir des majorants du facteur compétitivité d'algorithmes non clairvoyants $A \circ B$ obtenus par composition d'un ordonnanceur de tâches et d'un ordonnanceur de processus. De plus, nous avons vu qu'il était facile à calculer.

Ce coefficient capture en quelque sorte, le *pire degré de parallélisabilité* d'un graphe de dépendances pour un algorithme donné, au sens qu'il mesure quel est le pire gaspillage possible pour une structure de tâche donnée.

En ce sens, l'étude du coefficient d'étalement d'un DAG pour un algorithme donné, peut aider à la décision de l'organisation d'une tâche avant sa mise-en-œuvre, en permettant aux décideurs d'éviter de les organiser suivant des structures pour lesquelles nous savons que le risque de gaspillage est élevé. Ceci est d'autant plus intéressant que le cadre très général de notre étude permet d'inclure les imprévus des situations réelles. Aussi nous pensons qu'au-delà du résultat de la compétitivité de $\textbf{LAPS}_\beta \circ \textbf{EQUI}$, ce paramètre a son propre intérêt.

Dans certains cas, ce coefficient permet d'obtenir également des *minorants* du facteur de compétitivité d'algorithmes non-clairvoyants en général. Julien Robert a également démontré dans sa thèse qu'en définissant certains types de tâche dites *remplisseuses*, il était possible de démontrer que certains de ces minorants du coefficient d'étalement peuvent se généraliser à tout algorithme non clairvoyant.

Nous avons vu également que ce coefficient pouvait varier d'un algorithme à l'autre (v. p. ex., les coefficients **EQUI** et **SPLIT** pour les arbres sortant d-réguliers). Il serait intéressant de mesurer l'influence de la structure du DAG seule sur ce coefficient, indépendamment de l'algorithme utilisé, pour voir si l'on pourrait définir ce coefficient sous la forme d'un paramètre de graphe, capturant exactement la parallélisabilité de la structure de dépendances.

6 Applications à l'optimisation des serveurs web

Du fait de sa généralité, le modèle d'Edmonds [93] est capable de modéliser des situations très différentes. En 2003, Edmonds, Datta et Dymond ont présenté, par une réduction à ce modèle, la première analyse du bon fonctionnement du protocole TCP sans aucune hypothèse sur les requêtes [96]. Auparavant, en 2002, Edmonds et Pruhs ont pu utiliser ce même modèle pour démontrer qu'**Equi** était également performant pour la *diffusion* de messages (*broadcast*, en anglais) [97]. C'est ce dernier résultat qui va nous intéresser ici.

6.1 Les protocoles de diffusion (broadcast)

Des études (v. p. ex. [46, 6]) ont démontré depuis plusieurs décennies, que la plupart des requêtes émises vers les serveurs HTTP se concentrent sur un très petit nombre de pages (v. l'introduction de ma thèse [277] pour un état de l'art de ces études datant de 2000). Ceci pose deux problèmes : d'une part, ces pages sont diffusées un très grand nombre de fois et occupent donc une part très importante de la bande passante disponible ; d'autre part, le nombre de requêtes pour ces pages est extrêmement important et peut conduire à submerger le serveur qui risque de s'écrouler. Cette situation est particulièrement critique pour les sites web des grands événements sportifs, où un très grand nombre d'utilisateurs cherchent à accéder à un très petit nombre de pages avec une très grande fréquence, pour se tenir informés des tous derniers résultats.

Pour résoudre ces deux problèmes, il existe une solution simple qui est celle retenue par la télévision, la radio et le vidéotexte depuis fort longtemps : dédier un médium naturellement diffusant (câble ethernet, wifi, satellite, radio, télévision câblée, multicast par Internet, etc.) à la diffusion des pages et des informations les plus populaires. Les clients qui souhaitent recevoir ces informations, se connectent à ces canaux et attendent la diffusion des informations qui les intéressent, chaque diffusion étant téléchargeable simultanément par tous ceux qui l'ont demandée. Ces protocoles existent en deux variantes :

- *Push-based* (pseudo-interactive) où les utilisateurs n'émettent aucune requête. Le serveur diffuse en boucle les informations suivant un ordonnancement fixé à l'avance, précalculé à l'aide des statistiques des demandes des clients [15, 29, 212, 83, 277]). Ce protocole a l'avantage de pouvoir supporter un nombre arbitrairement élevé de clients sans risque de surcharge puisqu'aucune requête n'est émise. Ces performances sont totalement indépendantes du nombre de clients.
- *Pull-based* (à la demande) où les utilisateurs émettent une requête pour indiquer au serveur les informations qui souhaitent recevoir [7, 246, 97]. Le serveur doit alors calculer au vol la bande passante qu'il attribue à la diffusion de chaque information en fonction des demandes en attente des clients. Ce système a l'avantage d'être extrêmement flexible et réagit bien plus vite que le précédent lorsque le nombre de clients est faible, mais ses performances se dégradent rapidement et le serveur risque la surcharge si le nombre de clients augmente trop.

J'ai étudié intensivement ces protocoles (et plus particulièrement le premier) à l'époque de ma thèse [210, 278, 212, 277, 211] et on peut constater neuf ans après que si ces technologies ont trouvé leur place dans les intranet ou les réseaux par satellites, ils n'ont pas pris du tout à l'échelle d'Internet. La raison en est sans doute, premièrement, le déploiement rapide des fibres optiques qui ont permis une augmentation rapide des débits et des temps de réponses, et ensuite, le perfectionnement des stratégies de duplication des contenus à proximité des sources

qui permettent de répartir la charge localement (p. ex. les technologies Akumai). Le développement des technologies de *broadcast* et *multicast* a aussi été freiné par l'absence quasi-généralisée de l'implémentation des protocoles correspondants dans les routeurs d'Internet. L'implémentation la plus convaincante des protocoles *multicast* est actuellement le protocole pair-à-pair `BitTorrent` [70], un protocole qui a soulagé considérablement la charge induite par les échanges pair-à-pair sur Internet. Il n'en reste pas moins qu'avec la montée en puissance de la vidéo à la demande, ces protocoles pourraient bien connaître un renouveau d'intérêt à de plus grandes échelles.

Ceci étant dit, il s'agit également (et avant tout) d'un sujet particulièrement intéressant du point de vue algorithmique et encore mal compris : le statut de l'approximabilité du problème de la diffusion hors-ligne de fichiers de taille unitaire sans dépendance est toujours inconnu (le meilleur algorithme hors-ligne actuellement est une $\frac{\log^2 n}{\log \log n}$-approximation par [28]) et sa preuve de NP-complétude est elle-même très récente [103, 62].

6.2 Importance de dépendances entre les requêtes

Dans la conclusion de ma thèse où j'avais étudié intensivement l'optimisation des protocoles *push-based*, j'avais fait la remarque que sur Internet, on ne demande *jamais* un seul fichier, mais toujours un ensemble d'objets. En effet, chaque page HTML est composée d'un grand nombre d'éléments (logos, feuille de style, image de fond, barre de titre, index, etc.) qui sont pour la plupart partagés entre de nombreuses pages. Je posais alors la question du bien-fondé d'un modèle où les requêtes portent sur un objet à la fois. Nous proposons donc ici d'explorer des modèles où chaque utilisateur demande non pas un seul objet, mais un ensemble d'éléments, éventuellement partagés avec d'autres utilisateurs.

6.2.1 Importance de la prise en compte des dépendances

Il se trouve que la qualité de service d'un serveur web se dégrade considérablement si la diffusion des objets se fait en négligeant ces dépendances, comme c'est le cas actuellement sur Internet où chaque élément est diffusé chaque fois qu'une page qui le contient est demandée en ne tenant pas compte des différents éléments qui composent la requête. Nous l'avons démontré dans le cadre des protocoles *pushed-based* avec Sandeep Dey dans [83] (alors étudiant de M2 sous ma direction), puis dans le cadre *pull-based* avec Julien Robert [265]. Dans les deux cas, nous avons démontré qu'ignorer les dépendances peut conduire à une dérive d'un facteur multiplicatif $\sqrt{n}$ du temps de service par rapport à la stratégie optimale (où n désigne le nombre d'objets à diffuser). Il est donc crucial de se poser la question des dépendances pour l'optimisation de la diffusion de données.

6.2.2 Prise en compte des dépendances dans les protocoles pseudo-interactifs (push-based)

Les dépendances entre les requêtes avaient fait l'objet de différentes études, dès 1987 avec [14], puis dans [31] où le cas de deux éléments était complétement résolu. On peut aussi citer [30] où les performances d'une permutation aléatoire sont étudiées pour différents types de dépendances. Avec Sandeep Dey, nous avons proposé différents algorithmes d'approximation déterministe ou randomisé pour la diffusion d'ensembles d'objets de taille arbitraire, découpés en paquets de taille unitaire. Le modèle est alors le suivant. On dispose d'un ensemble de n éléments. Chaque requête demande un sous-ensemble S des éléments, suivant une

distribution p_S à support polynomial (c.-à-d., $p_S = 0$ sauf sur un nombre polynomial de sous-ensembles). Les requêtes arrivent à des dates déterminées par un processus de Poisson (c.-à-d., à des instants uniformes sur tout intervalle fini), demandant chacune l'ensemble S de fichiers indépendamment avec probabilité p_S. Il s'agit alors pour le serveur de trouver un ordonnancement périodique minimisant l'espérance temps de service moyen d'un client aléatoire suivant la loi décrite ci-dessus. Nous avons obtenu en 2006 la première approximation polynomiale à facteur constant pour ce problème :

Théorème 3.24 (Diffusion pseudo-interactive d'ensemble d'éléments, [83]). *Il existe une 4-approximation polynomiale et déterministe pour la diffusion pseudo-interactive (push-based) d'éléments.*

Cet algorithme repose sur une formulation du problème sous la forme d'un programme non-linéaire convexe que l'on résout par méthode des ellispoïdes [154]. Nous ne détaillerons pas plus en avant ce résultat dans ce manuscrit et renvoyons le lecteur à l'article [83].

6.2.3 Prise en compte des dépendances dans les protocoles à la demande (pull-based)

À la suite des résultats encourageants que nous avions obtenus avec Sandeep Dey, nous avons démarré avec Julien Robert durant son stage de M2, l'étude de requêtes ensemblistes dans le cadre du modèle *pull-based* proposé par Edmonds et Pruhs dans [97]. Nous avons démontré dans [265] l'importance de baser ses décisions sur les requêtes et non sur les objets demandés uniquement (comme c'est le cas dans les protocoles actuels). Nous avons alors introduit l'algorithme **Equi** $\circ$ **Equi** dont nous avons prouvé qu'il est $(4+\varepsilon)$-rapide $O(1)$-compétitif pour tout $\varepsilon > 0$. La preuve était alors relativement compliquée mais se simplifie considérablement avec nos travaux postérieurs [266] présentés à la section précédente. Après une présentation du modèle et un bref rappel historique, la suite de cette section est consacrée à la présentation de ce résultat.

6.3 Modèle de diffusion à la demande (pull-based) avec dépendances

Instance du problème. Une instance consiste en la donnée de :

- un ensemble $\mathcal{I}$ de m fichiers $I_1, \ldots, I_m$ de longueurs $\ell_1, \ldots, \ell_m$, supposées inconnues de l'ordonnanceur.[49]
- un ensemble $\mathcal{S}$ de n requêtes pour n ensembles non-vides de fichiers $S_1, \ldots, S_n \subseteq \mathcal{I}$ arrivant aux dates $r_1, \ldots, r_n$, découvertes par l'ordonnanceur au moment de leurs arrivées et pas avant.

Ordonnancement. Un ordonnancement $\mathcal{B}_s$ sur une bande passante s est un ensemble de fonctions positives, constantes par morceaux $\rho_j : t \mapsto \rho_j^t$, avec $1 \leqslant j \leqslant m$, telles que pour tout t,

$$\sum_j \rho_j^t \leqslant s.$$

[49] Comme nous n'imposons aucune limite sur m, on peut même supposer que la taille des fichiers est imprévisibles et évolue au court du temps au gré des mises-à-jour et des informations disponibles. Il suffit pour cela de créer un nouveau fichier avec un nouvel indice pour chaque nouvelle version du fichier.

Chaque ρ_j^t ($\in \mathbb{R}_+$) représente la bande passante allouée à la diffusion du fichier I_j au temps t. Dans ce modèle préemptif, il est possible d'allouer une fraction arbitraire de processeur à chaque processus, ce qui, en pratique est réalisé par *time multiplexing*.[50]

En posant $c_j^0 = 0$, nous disons que la $k^{\text{ème}}$ diffusion de I_j se termine à la date c_j^k où c_j^k est le premier instant t tel que la bande passante totale allouée à I_j depuis $t = c_j^{k-1}$ vaut ℓ_j :

$$c_j^k = \min \left\{ t : \int_{c_j^{k-1}}^{t} \rho_j^t \, dt \geqslant \ell_j \right\}$$

On note b_j^k, la date du début de la $k^{\text{ème}}$ diffusion de I_j :

$$b_j^k = \inf\{t \geqslant c_j^{k-1} : \rho_j^t > 0\}$$

Remarquons que dans ce modèle nous n'autorisons ni la diffusion simultanée du même fichier ni l'abandon d'une diffusion en cours. Les multi-diffusions sont en effet inefficaces puisqu'il est plus avantageux de consacrer la totalité de la bande passante allouée à ces diffusions pour d'abord diffuser le fichier une première fois, puis de le diffuser une seconde fois ensuite. Quant à l'abandon en cours d'une diffusion, c'est au strict désavantage de notre algorithme puisqu'un ordonnancement optimal ne commencera jamais une diffusion pour l'abandonner ensuite.

Pour chaque date t, on note B_j^t et C_j^t les dates du début et de fin de la première diffusion du fichier I_j après t :

$$B_j^t = \min\{b_j^k : b_j^k \geqslant t\} \quad \text{et} \quad C_j^t = \min\{c_j^k : b_j^k \geqslant t\}$$

Fonction objectif. Le coût d'un ordonnancement adopté ici est le *temps de service moyen* des n utilisateurs. Comme [97], nous considérons qu'il est impossible de commencer à télécharger un fichier au mileu de sa diffusion, ce qui est le choix le plus courant des protocoles actuels.[51] Chaque requête S_i arrivant à la date r_i, télécharge donc chaque fichier $I_j \in S_i$ entre $B_j^{r_i}$ et $C_j^{r_i}$. La requête est servie lorsqu'elle a téléchargé tous ses fichiers, et son temps de complétion c_i est donc :

$$c_i = \max_{I_j \in S_i} C_j^{r_i}$$

Nous dirons qu'une requête est *active* ou *en vie* entre r_i et c_i. Nous cherchons à minimiser le *temps de flot moyen* des requêtes, $\frac{1}{n}\sum_{i=1}^{n}(c_i - r_i)$, ou de manière équivalente du point de vue de l'approximation, le *temps de flot* de l'ordonnancement :

$$\text{BFlowtime}(\mathcal{B}_s) = \sum_{i=1}^{n}(c_i - r_i)$$

On note $\text{BOPT}_s(\mathcal{I}, \mathcal{S}) = \min_{\mathcal{B}_s} \text{BFlowtime}(\mathcal{B}_s)$ le temps de flot optimal pour l'ensemble de requêtes $\mathcal{S}$ en disposant d'une bande passante s.

[50] Notons que [97] propose une technique pour limiter le nombre de préemptions à une seule en espérance pour chaque fichier. Nous ne présenterons pas ici cette technique que nous avons utilisée dans [265], mais il est intéressant de savoir qu'elle existe.

[51] Contrairement au modèle de [97], nous pouvons aisément contourner cette restriction dans notre modèle, en découpant chaque fichiers en paquets : pour demander un fichier, on demande l'ensemble de ses paquets que l'on peut télécharger indépendamment.

Non-clairvoyance et Compétitivité. De même que précédemment, nous ne considérerons que des algorithmes *non clairvoyants* ignorant tout des fichiers qu'ils diffusent et des requêtes avant leurs arrivées. Les seules informations dont dispose l'algorithme pour établir son ordonnancement sont le nombre et les temps de vie des requêtes actives. Nous noterons $A_s(\mathcal{I}, \mathcal{S})$ l'ordonnancement calculé par A pour l'instance $(\mathcal{I}, \mathcal{S})$ sur une bande passante s.

Ce modèle de diffusion est clairement une généralisation du problème de l'ordonnancement de tâches consistant en un ensemble de processus traversant une unique phase **PAR**, étudié dans les sections précédentes. Ce dernier correspond au cas où tous les clients demandent des fichiers distincts de ceux demandés par les autres clients. Les minorants de [224] s'appliquent donc et il n'existe aucun algorithme non clairvoyant c-compétitif pour $c < \sqrt[3]{n}$ s'il est déterministe et $c = \omega(\log n)$ s'il est randomisé. En fait pour la diffusion de fichiers de taille unitaire en l'absence de dépendance (chaque requête demande un fichier à la fois), [182] et [27] démontrent respectivement qu'aucun algorithme en-ligne déterministe ne peut être $o(n)$-compétitif, et qu'aucun algorithme en-ligne randomisé ne peut être $o(\sqrt{n})$-compétitif.

Nous recourons donc de nouveau aux techniques d'analyse par augmentation de ressources. Nous dirons qu'un algorithme de diffusion A est s-rapide c-competitif si pour toute instance $(\mathcal{I}, \mathcal{S})$,

$$\text{BFlowtime}(A_s(\mathcal{I}, \mathcal{S})) \leqslant c \cdot \text{BOPT}_1(\mathcal{I}, \mathcal{S})$$

6.4 Résultats connus en l'absence de dépendance

Edmonds et Pruhs ont proposé en 2002 [97] le premier algorithme compétitif pour la diffusion de fichiers en ligne, dans le cas où chaque utilisateur ne demande qu'un unique fichier à la fois. Le principe de leur algorithme **BEQUI** est de mimer le comportement d'**EQUI** sur une instance de tâches (du type des sections précédentes) construite à la volée par **BEQUI**. Nous verrons dans la section suivante en quoi consiste ce processus de mime. La clé de leur analyse est alors de décider des phases de ces tâches *a posteriori* pour garantir que l'on puisse comparer les performances de $\textbf{BEQUI}_s$ à BOPT_1 en utilisant le résultat d'Edmonds [93] sur l'analyse d'**EQUI**.

Le schéma général de leur preuve est le suivant. Ils démontrent que la quantité de travail **PAR** choisie pour l'instance de tâches ne dépasse pas le double de celle des fichiers diffusés. Il est ainsi possible d'obtenir un ordonnancement valide des tâches sur deux processeurs, en étirant verticalement d'un facteur 2 l'ordonnancement optimal des fichiers originaux sur bande passante 1. Le temps de flot des tâches dans l'ordonnancement obtenu est alors identique au temps flot optimum des fichiers sur bande passante 1. D'autre part, $\textbf{BEQUI}_s$ et $\textbf{EQUI}_s$ ont le même temps de flot par construction. On sait par [93] que $\textbf{EQUI}_s$ est s-rapide $O(1)$-compétitif pour tout $s > 2$. Ainsi, pour tout $s > 4$, le temps de flot d'$\textbf{EQUI}_s$ sur l'instance de tâches construite à la volée par **BEQUI** est à un facteur constant de celui de l'ordonnancement sur deux processeurs de ces mêmes tâches. Or le temps de flot de cet ordonnancement est inférieur au temps de flot des fichiers sur bande passante 1. Il s'en suit que pour tout $s > 4$, le temps de flot de $\textbf{BEQUI}_s$ est compétitif par rapport à l'optimum sur bande passante 1. Ainsi,

Théorème 3.25 (cas sans dépendance, [97]). *Il existe un algorithme non clairvoyant,* **BEQUI**, *s-rapide $O(1)$-compétitif pour tout $s > 4$*

Leur réduction au problème d'ordonnancement de tâches étant complètement modulaire, leur analyse de $\textbf{LAPS}_\beta$ permet de conclure en mimant $\textbf{LAPS}_\beta$ à la place d'**EQUI** que :[52]

[52] Nous verrons que cette conséquence n'est pas aussi directe que semble l'indiquer [99].

Algorithme 3.4 Ordonnanceur de diffusions BA mimant un ordonnanceur de tâches A

entrée(s) : une bande passante s, un ensemble $\mathcal{I}$ de m fichiers $I_1, \ldots, I_m$, une instance $\mathcal{S}$ de n requêtes pour des ensembles $S_1, \ldots, S_n \subseteq \mathcal{I}$ arrivant aux dates $r_1, \ldots, r_n$, et A un ordonnanceur non clairvoyant de tâches avec contraintes de dépendances du type de la section 2.

Poser $i := 0$
Démarrer l'exécution de l'algorithme A sur s processeurs.
pour chaque instant t **faire**
  pour chaque requête pour un ensemble S de fichiers arrivant à l'instant t **faire**
    Soient $I_{j_1}, \ldots, I_{j_q}$ les fichiers demandés par S.
    Faire $i := i + 1$, poser $S_i := S$, puis créer une nouvelle tâche J_i consistant en q processus indépendants $J_{ij_1}, \ldots, J_{ij_q}$ arrivant à $r_i := t$ et informer A de l'arrivée de la tâche J_i.
  pour chaque fichier I_i dont la diffusion, démarrée à un instant $B_i < t$, se termine à l'instant t **faire**
    pour chaque (i, j) tel que la tâche J_i est arrivée à $r_i \leqslant B_i$ et contient un processus J_{ij} **faire**
      Annoncer à A que le processus J_{ij} de la tâche J_i est maintenant terminé.
  Soit ρ_{ij}^t la quantité de processeurs attribuée par A_s à chaque processus en vie J_{ij}.
  Attribuer une bande passante $\rho_j^t = \sum_{\text{i: processus } J_{ij} \text{ est en vie}} \rho_{ij}^t$ à chaque fichier I_j.

Théorème 3.26 (cas sans dépendance, "[97]+[99]×amélioration de la section 4"). *Il existe un algorithme non clairvoyant,* **BLAPS**$_\beta$*, s-rapide $\frac{2s}{\beta\varepsilon}$-compétitif pour tout $s > 2(1+\beta)+\varepsilon$.*

Nous reprendrons cette étude en prenant notre temps dans la suite de cette section.

Plus récemment. Toujours sans dépendance et dans le cas où les fichiers ont tous la même longueur, ils proposèrent une première analyse de l'algorithme le plus utilisé en pratique en *broadcast*, **LWF** pour *Longest Wait First*, où l'on diffuse à chaque instant le fichier pour lequel l'attente cumulée des clients est maximale. Dans [98], ils démontrent que cet algorithme est 6-rapide $O(1)$-compétitif mais n'est pas s-rapide $O(1)$-compétitif pour tout $s < \frac{1+\sqrt{5}}{2}$. Ce résultat a été amélioré tout récemment par [65] qui démontrent que **LWF** est en fait $(3.4+\varepsilon)$-rapide $O(1/\varepsilon^3)$-compétitif lorsque tous les fichiers sont de longueur 1.

Notre résultat. Nous présentons ici une application du théorème 3.15 de la section précédente pour obtenir un algorithme compétitif pour la diffusion *(broadcast)* d'ensembles de fichiers. Historiquement, nous avons obtenu ce résultat avant ceux présentés à la section précédente. C'est effectivement dans [265], que nous avons introduit la notion d'algorithmes $A \circ B$. Il est cependant maintenant bien plus commode de présenter ce résultat comme une conséquence directe de [266] et [97, 99].

6.5 Algorithmes mimes

Nous nous plaçons à présent dans le modèle général défini à la section 6.3.

Algorithmes BA. Nous généralisons l'algorithme proposé par Edmonds et Pruhs dans [97] pour le cas où chaque requête ne demandait qu'un fichier à la fois, comme suit. La description

complète de notre algorithme est donnée à l'algorithme 3.4.

On se fixe un ordonnanceur A non clairvoyant de *tâches* avec contraintes de dépendance. On démarre l'exécution de A sur s processeurs. À l'arrivée de la $i^{\text{ème}}$ requête $S_i = \{I_{j_1}, \ldots, I_{j_q}\}$ à $t = r_i$, on crée une tâche J_i constituée de q processus indépendants $J_{ij_1}, \ldots, J_{ij_q}$, un par fichier demandé ; et on informe A de l'arrivée de cette tâche (et donc de ses q processus). Chaque processus J_{ij} reste en vie jusqu'à la fin de la prochaine diffusion du fichier j qui aura lieu à $t = C_j^{r_i}$, date à laquelle le processus J_{ij} sera déclaré terminé à A. Ceci assure que le temps de flot de la tâche J_i est bien égal au temps de flot de la requête à laquelle elle est associée : la tâche termine quand ses q processus terminent, c.-à-d. quand les q fichiers demandés ont été diffusés. À chaque instant, l'algorithme de diffusion, noté BA_s, disposant d'une bande passante s, simule l'algorithme A_s sur s processeurs pour l'instance de tâches $\{J_{ij}\}$ construite à la volée, et accorde pour la diffusion chaque fichier I_j une bande passante $\rho_j^t = \sum_{j:J_{ij} \text{ est en vie}} \rho_{ij}^t$, où ρ_{ij}^t est la quantité de processeurs attribuée par A au processus J_{ij} de la tâche J_i à l'instant t.

L'astuce est que : comme A est non clairvoyant, il est *inutile* de définir les phases que traversent les processus J_{ij} pour décrire le fonctionnement de l'algorithme ! A fait ce qu'il a à faire et BA ne fait que le mimer. Ces phases seront fixées *a posteriori* dans l'analyse *uniquement* et *comme cela nous arrange* pour démontrer que la compétitivité de BA est héritée de celle de A, c.-à-d. que BA est $2s$-rapide c-compétitif pour la diffusion d'ensembles de fichiers, si A est lui-même s-rapide c-compétitif pour l'ordonnancement de tâches avec dépendances et *continu à droite* (une condition nécessaire, "oubliée" dans [97]).

6.6 Fixer les phases des processus

L'algorithme A utilisé par BA étant non clairvoyant, nous avons entière latitude pour fixer les phases des processus de la façon qui nous convient le mieux pour analyser l'algorithme BA. Nous allons nous arranger pour que ces phases nous permettent de comparer facilement le temps de flot de $BA_{2s}(\mathcal{I}, \mathcal{S})$ à $\text{BOPT}_1(\mathcal{I}, \mathcal{S})$ en passant par $\text{OPT}_2(J)$.

Soient $\mathcal{I} = \{I_1, \ldots, I_m\}$ un ensemble de m fichiers de longueurs $\ell_1, \ldots, \ell_m$ (inconnues de l'ordonnanceur) et $\mathcal{S}$ une suite de m requêtes pour des ensembles $S_1, \ldots, S_n \subseteq \mathcal{I}$ arrivant à des dates $0 \leqslant r_1 \leqslant \ldots \leqslant r_n$. Soient $\mathcal{B}_s$ et $\mathcal{S}_s$ les ordonnancements de fichiers et de tâches calculés simultanément par BA_s et A_s sur bande passante s et s processeurs respectivement. Soient $\{J_{ij}\}$ l'instance de tâches construite à la volée par BA_s pour la soumettre à A_s. Les seules caractéristiques de chaque processus J_{ij} qui soient fixées pour l'instant sont : sa date d'arrivée, égale à la date d'arrivée r_i de la $i^{\text{ème}}$ requête S_i, et sa date de complétion, égale à $C_j^{r_i}$, la date de la fin de la première diffusion du fichier I_j après r_i dans $\mathcal{B}_s$.

Considérons $\mathcal{BO}_1$ un ordonnancement des diffusions des fichiers $I_1, \ldots, I_m$ sur une bande passante 1 qui soit quasi-optimal pour les requêtes $S_1, \ldots, S_n$, c.-à-d. tel que $\text{BFlowtime}(\mathcal{BO}_1) \leqslant \text{BOPT}_1(\mathcal{I}, \mathcal{S}) + \eta$ pour un $\eta > 0$ arbitrairement petit.

Nous décidons de donner deux phases à chaque processus, une **Seq** suivie d'une **Par** que nous définissons comme suit. Le processus J_{ij} est créé par la requête S_i arrivant à la date r_i et demandant le fichier I_j parmi d'autres. Soient a et b les dates du début et de la fin de la première diffusion du fichier I_j après la date r_i dans $\mathcal{B}_s$, et a', la date du début de la première diffusion dans $\mathcal{BO}_1$ du fichier I_j après la date r_i. La requête S_i commence à télécharger le fichier I_j à la date a dans $\mathcal{B}_s$ et à la date a' dans $\mathcal{BO}_1$, et termine de le télécharger à la date b dans $\mathcal{B}_s$. Nous fixons alors les trois phases de J_{ij} ainsi :

- la phase **Seq** de travail $\min(b, a') - r_i$, s'étend de $t = r_i$ à $t = \min(b, a')$, c.-à-d.

jusqu'à ce que la première diffusion de I_j après $t = r_i$ commence dans $\mathcal{BO}_1$ ou bien finit dans $\mathcal{B}_s$.

- Si $b > a'$, elle est suivie d'une phase **PAR** de travail total $w_{ij} = \int_{a'}^{b} \rho_{ij}^t \, dt$, c.-à-d. égale à la quantité totale de processeurs attribuée à ce processus de $t = a'$ à la complétion de J_{ij} par l'ordonnanceur de tâches A_s.

Ces deux phases sont calculées pour rentrer exactement dans l'ordonnancement A_s. Pourtant, elles ne correspondent pas tout à fait à ce que l'on désire, car il est possible que A_s décide *autoritairement* de ne pas ordonnancer la tâche J_{ij} (comme **LAPS**$_\beta$ peut le faire p. ex., en privilégiant des processus issus de requêtes plus récentes pour le même fichier I_j). Ainsi, si on note $c = \inf\{t \leqslant b : \rho_{ij}^t = 0\}$ le dernier instant où J_{ij} se voit attribuer des processeurs par A_s avant sa complétion "officielle", et si on s'en tient à la définition de la section 2 du temps de flot, le temps de flot processus sera $c - r_i$ et non $b - r_i$ comme nous le souhaitons ; pire, l'algorithme A_s risque de ne pas calculer le bon ordonnancement puisque l'ultime phase **PAR** de J_{ij} étant terminée à $t = c < b$, la tâche J_{ij} ne sera plus active entre c et b, alors qu'elle est supposée l'être dans l'algorithme BA_s.

Edmonds et Pruhs proposent dans [97] de *« rajouter une quantité de travail infinitésimale à la phase* **PAR** *»* afin que le processus J_{ij} reste en vie au-delà de c. Cette proposition pose plusieurs problèmes : il faut que A_s ordonnance de nouveau J_{ij} pour terminer cette quantité infinitésimale, ce qui risque de n'arriver que bien après b ; l'ordonnancement risque donc d'être perturbé par la suite par ce processus encore vivant alors qu'officiellement il ne doit plus l'être. Si cette astuce fonctionne pour **EQUI** (où le problème ne se pose pas vraiment puisque tout processus en vie est ordonnancé), il est impossible de la généraliser à tout algorithme comme le propose le théorème 3 de [97]. En particulier, comme nous verrons sur un exemple à la section 6.8, il est tout à fait possible de rajouter une quantité de travail infinitésimale à tous les processus terminés et que cela *diminue* dans **LAPS** d'une valeur arbitrairement grande le temps de flot d'une proportion constante de tâches au lieu de l'augmenter ! Nous verrons en particulier que l'ajout d'un nombre fini de quantités de travail infinitésimales peut perturber l'ordonnancement sur un intervalle de temps arbitrairement long, ce qui fait s'effondrer l'argumentation développée dans [97].

Plutôt que de recourir au travail infinitésimal, nous préférons introduire dans ce manuscrit un nouveau type de phase, la phase *zombie*. Un processus peut entrer dans une phase *zombie* (**ZOMB**) de durée ℓ si c'est sa dernière phase : il reste en vie durant toute cette phase qui débute immédiatement après la précédente et dure un temps ℓ, à moins que l'ordonnanceur ne tente de l'ordonnancer avant sa fin, auquel cas la phase et le processus sont immédiatement déclarés terminés. Nous rajoutons donc à chaque processus J_{ij} une phase **ZOMB** de durée $b-c$ à la suite de sa phase **PAR**. Notons que l'ordonnancement optimum n'est pas modifié par la présence des phases zombie puisqu'il lui suffit de poser ponctuellement $\rho_{ij}^t = 1$ à un instant t séparé d'au plus $\frac{\eta}{m2^i}$ de la terminaison de la phase précédente pour s'en débarrasser (le surcoût est d'au plus η sur le temps flot total).

On désigne désormais par ZFlowtime($\mathcal{S}_s$) le temps de flot d'un ordonnancement $\mathcal{S}_s$ dont certains processus finissent par une phase zombie.

Fait 3.27. *Par construction,* BFlowtime($\mathcal{B}_s$) = ZFlowtime($\mathcal{S}_s$).

Compétitivité de Laps$_\beta \circ \alpha$-étaleur en présence de processus zombie. La compétitivité de **LAPS**$_\beta \circ \alpha$-étaleur repose sur le lemme 3.14 qui démontre que pour $s > 1 + \beta + \varepsilon$, $n_t + \frac{d\Phi(t)}{dt} \leqslant \frac{2s}{\beta\varepsilon} \cdot \ell_t$ à tout instant t qui ne correspond ni à une terminaison ni à une arrivée de

tâche. La démonstration de ce lemme repose sur une majoration des variations de la fonction potentiel Φ. Or, si l'on regarde la majoration utilisée pour la contribution (négative) de **LAPS**$_\beta$ à cette variation, on constate que seuls les processus *ordonnancés* sont considérés : cette majoration reste donc valable si des processus *non-ordonnancés* sont en vie dans une phase finale **ZOMB**. Nous en déduisons le théorème suivant, corollaire direct du théorème 3.15.

Théorème 3.28. *Pour tout ordonnanceur de processus α-étaleur B,* **LAPS**$_\beta \circ B$ *est s-rapide $\frac{2s\alpha}{\beta\varepsilon}$-compétitif pour toute instance de tâches constituées de DAG arbitraires de processus incluant des phases* **ZOMB**.

6.7 Compétitivité de LAPS$_\beta$ ∘ affairé

Pour conclure la preuve, il suffit maintenant de démontrer, en suivant le schéma [97], que : $\text{Flowtime}(\mathcal{BO}_{[2/1]}[J/\mathcal{S}]) \leqslant \text{BFlowtime}(\mathcal{BO}_1)$, c.-à-d., que si l'on étire verticalement l'ordonnancement $\mathcal{BO}_1$ d'un facteur 2, on peut y ordonnancer les tâches J_i que nous avons créées ; ensuite, nous conclurons en démontrant que tout ordonnanceur de processus *affairé*, c.-à-d. qui utilise tous les processeurs qu'on lui donne, est un 1-étaleur pour les tâches **SEQ**-puis-**PAR**-puis-**ZOMB**.

Lemme 3.29 ([97, 265, 263]). *Il existe un ordonnancement $\mathcal{O}_2$ sur deux processeurs des tâches $J_1, \ldots, J_n$ tel que :*

$$\text{ZFlowtime}(\mathcal{O}_2) \leqslant \text{BFlowtime}(\mathcal{BO}_1)$$

Démonstration. Nous reprenons ici les arguments de [97] en les simplifiant considérablement. Considérons un fichier I_j et plaçons tous les processus J_{ij} qui lui sont associés. Considérons deux diffusions consécutives du ficher I_j débutant à $t = a'$ et $t = a''$ ($a' < a''$) dans $\mathcal{BO}_1$ (on considère par convention que I_j a été diffusé par $t = 0^-$). Considérons l'ensemble des requêtes S_i contenant I_j arrivées entre a' et a''. Elles téléchargeront toutes I_j dans $\mathcal{BO}_1$ lors de la diffusion qui débute en $t = a''$. Considérons à présent la première diffusion de I_j qui a lieu dans $\mathcal{B}_s$ juste après $t = a''$, à $t = a \geqslant a''$.

Chaque requête S_i contenant I_j arrivée entre $t = a'$ et $t = a''$ engendre dans l'algorithme 3.4 un processus J_{ij} dont la phase **SEQ** s'étend au plus jusqu'à $t = a''$ et si tel est le cas, admet une phase **PAR** dont le travail correspond par construction à sa contribution à la bande passante utilisée dans BA_s pour terminer la diffusion en cours de I_j avant $t = a$ puis effectuer la diffusion suivante, démarrant à $t = a$ jusqu'à $t \leqslant b$ (éventuellement suivi d'une phase **ZOMB** dont on se débarrassera sans surcoût).

Il s'en suit que la quantité totale de travail **PAR** de ces tâches est inférieur à $2\ell_j$ et est immédiatement disponible à partir de $t = a''$. Il est donc possible d'ordonnancer toutes ces tâches J_{ij} créées par les requêtes demandant I_j arrivant entre $t = a'$ et $t = a''$ durant le créneau réservé à la diffusion de I_j démarrant à $t = a''$ dans $\mathcal{BO}_1$, après l'avoir étiré verticalement d'un facteur 2. On se débarrasse enfin des phases **ZOMB** restantes sans surcoût en attribuant ponctuellement $\frac{2}{n}$ processeurs à la date $t = a''$ à chacun d'eux.

En procédant ainsi pour tous les fichiers et tous les intervalles entre deux diffusions consécutives de ce fichier dans $\mathcal{BO}_1$, on obtient ainsi un ordonnancement $\mathcal{O}_2$ sur deux processeurs de l'ensemble des tâches créées par l'algorithme BA_s durant son exécution, dont le temps de flot est inférieur à celui de $\mathcal{BO}_1$. ■

Ordonnanceurs de processus affairés. Nous dirons qu'un ordonnanceur de B est *affairé* s'il distribue entre les processus actifs de la tâche qu'il traite, la totalité des processeurs attribuée à cette tâche par l'ordonnanceur de tâches, c.-à-d. s'il ne gâche pas *volontairement* les processeurs qu'on lui donne. Tout algorithme raisonnable est affairé. Remarquons que nous ne faisons aucune hypothèse sur l'ordre ou la façon dont il traite les processus actifs de la tâche.

Lemme 3.30 ([265, 263]). *Tout ordonnanceur de processus affairé est un* 1*-étaleur pour les tâches constituées de processus* **SEQ***-puis-***PAR***-puis-***ZOMB** *indépendants.*

Démonstration. Considérons une tâche J constituée de processus **SEQ**-puis-**PAR**-puis-**ZOMB** indépendants. Remarquons qu'il est impossible de gâcher des processeurs sur une phase **ZOMB** puis qu'elle se termine dès qu'on essaye de l'ordonnancer. Les seules sources de gâchis sont donc les phases **SEQ** initiales des processus. Or, comme ces processus sont indépendants, quelle que soit l'allocation des processeurs, ces phases termineront toutes au bout d'un temps égal à la longueur de la plus longue d'entre elles. Après $t = \textsc{Seq}(J)$, tous les processus actifs seront dans une phase **PAR** ou **ZOMB** et comme l'ordonnanceur est affairé, aucun processeur ne sera gâché. Le gâchis de cet algorithme est donc bien $\leqslant 1 \cdot \textsc{Seq}(J)$. Son coefficient d'étalement vaut donc 1. ■

Théorème 3.31 ([265, 263]). *Pour tout ordonnanceur de processus B, l'ordonnanceur de diffusion non clairvoyant* $\textsc{Blaps}_\beta \circ B$ *(l'algorithme 3.4 avec* $A = \textsc{Laps}_\beta \circ B$*) est s-rapide $\frac{2s}{\beta\varepsilon}$-compétitif pour l'ordonnancement de la diffusion de fichiers avec dépendances pour tout $s > 2(1+\beta)+\varepsilon$.*

Démonstration. Notons $\mathcal{B}_s$ et $\mathcal{S}_s$ respectivement l'ordonnancement des diffusions des fichiers calculé par $\textsc{Blaps}_{\beta,s} \circ B$ et l'ordonnancement engendré par $\textsc{Laps}_{\beta,s} \circ B$ pour les tâches $J_1, \ldots, J_n$ créées à la volée. Soit $\mathcal{BO}_1$ un ordonnancement quasi-optimal de la diffusion des fichiers, tel que $\text{BFlowtime}(\mathcal{BO}_1) \leqslant \text{BOPT}_1 + \eta$ pour un $\eta > 0$ arbitrairement petit. Notons $\mathcal{O}_2$ l'ordonnancement sur deux processeurs des tâches $J_1, \ldots, J_n$ construit par le lemme 3.29 à partir de $\mathcal{BO}_1$. Nous avons :

$$
\begin{aligned}
\text{BFlowtime}(\mathcal{B}_s) &= \text{ZFlowtime}(\mathcal{S}_s) && \text{(fait 3.27)}\\
&\leqslant \frac{2 \cdot \frac{s}{2}}{\beta \cdot \frac{\varepsilon}{2}} \cdot \text{OPT}_2(J) && \text{(lemmes 3.28 et 3.30, et } s > 2(1+\beta)+\varepsilon)\\
&\leqslant \frac{2s}{\beta\varepsilon} \cdot \text{ZFlowtime}(\mathcal{O}_2) && (\mathcal{O}_2 \text{ est un ordonnancement valide des tâches})\\
&\leqslant \frac{2s}{\beta\varepsilon} \cdot \text{BFlowtime}(\mathcal{BO}_1) && \text{(lemme 3.29)}\\
&\leqslant \frac{2s}{\beta\varepsilon}(\text{BOPT}_1 + \eta), \quad \text{pour un } \eta > 0 \text{ arbitrairement petit.}
\end{aligned}
$$

■

Remarquons que nous avons en fait démontré que la compétitivité d'un algorithme d'ordonnancement de tâches compétitif avec des phases **ZOMB**, engendre un algorithme de diffusion de fichiers compétitif. D'où le théorème suivant, qui lève certaines imprécisions du théorème 3 de [97] :

Théorème 3.32. *Pour tout algorithme d'ordonnancement de tâches A s-rapide et c-compétitif en présence de contraintes de dépendances* et de phases **ZOMB***, l'algorithme 3.4 est* $2s$*-rapide c-compétitif.*

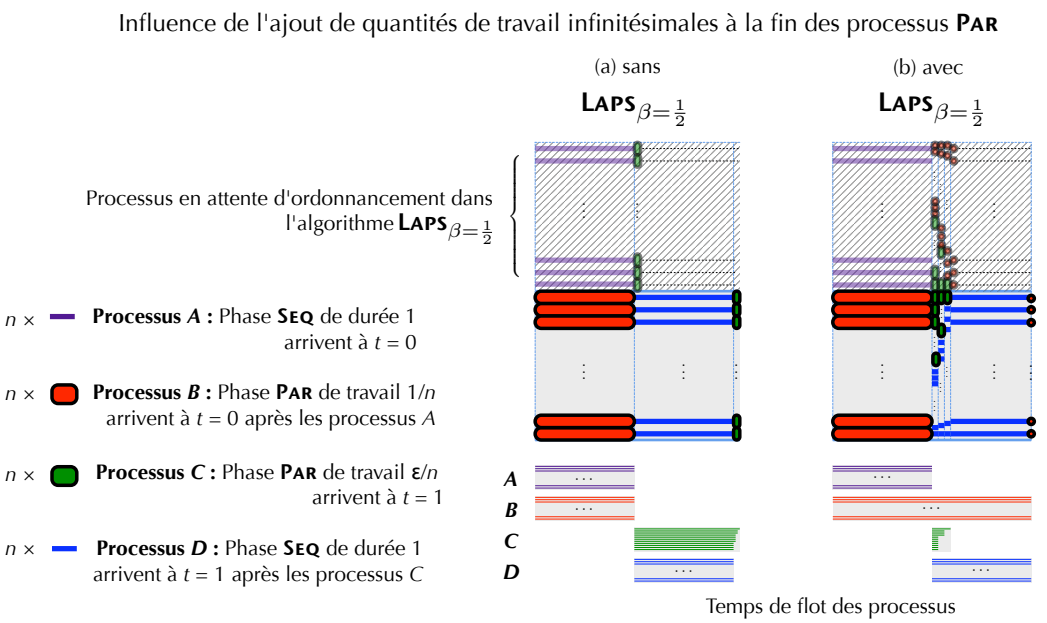


Figure 3.11 – Exemple de comportement non-monotone de LAPS$_\beta$: (a) l'ordonnancement calculé par LAPS$_{\frac{1}{2}}$ sur l'instance initiale ; (b) celui calculé calculé par LAPS$_{\frac{1}{2}}$ si une quantité de travail infinitésimale est ajoutée au bout des phases PAR des processus B et C.

6.8 Exemples de comportements inattendus de LAPS$_\beta$

Revenons sur la généralisation proposée du théorème 3 [97] du résultat sur la compétitivité de BEQUI démontrée dans ce même article. Les auteurs suggèrent que l'on peut démontrer que si A est un algorithme s-rapide c-compétitif pour l'ordonnancement de tâches (sans dépendance), alors l'algorithme BA est $2s$-rapide c-compétitif pour l'ordonnancement de diffusion (sans dépendance). Comme l'algorithme A_s peut décider volontairement (comme LAPS$_\beta$) de ne pas diffuser une tâche active (c'est même sa seule liberté), cela pourrait fausser la période d'activité de s processus qui pourraient être de fait terminés avant la fin de la diffusion du fichier dans l'ordonnancement calculé par BA : les temps de flots seraient alors incomparables. Pour y remédier, ils suggèrent d'« ajouter une quantité de travail PAR infinitésimale aux phases parallèles des tâches qui sont terminées de fait prématurément, *cela ne modifie pas l'ordonnancement A_s et assure que les temps de flot de chaque tâche soient identiques dans les deux ordonnancements* ». C'est la deuxième partie de cette assertion que nous contestons : ajouter une quantité infinitésimale de travail PAR peut perturber considérablement l'ordonnancement calculé par A sur des périodes arbitrairement longues, et même *raccourcir* arbitrairement le temps de flot des tâches que l'on a prolongé de cette quantité !

Une telle situation est illustrée par l'exemple de la figure 3.11 : on fixe $\beta = \frac{1}{2}$;[53] à $t = 0$, arrivent n tâches A en phase SEQ de durée 1 suivies de n tâches B en phase PAR de travail $\frac{1}{n}$ chacune ; à $t = 1$, arrivent n tâches C en phase PAR de travail $\frac{\varepsilon}{n}$ chacune (pour un $\varepsilon > 0$ arbitrairement petit), suivies de n tâches D en phase SEQ de durée 1. Nous étudions deux situations : (a) les tâches sont ordonnancées telles quelles et (b) on ajoute à la fin de chaque phase PAR une quantité infinitésimale de travail PAR. Dans (a), à $t = 1$ toutes les tâches A

[53] Cet exemple se généralise à toute valeur de $\beta > 0$.

et B terminent et les tâches C et D commencent, comme les D sont arrivées en dernier, ce sont elles qui sont ordonnancées d'abord par **LAPS**$_{\frac{1}{2}}$ et les tâches C ne sont ordonnancées qu'ensuite, entre $t = 2$ et $t = 2 + \varepsilon$. Imaginons à présent que l'on veuille prolonger la vie des tâches C et B en leur ajoutant une quantité infinitésimale de travail **PAR**. On observe alors dans (b) la chose suivante : les tâches B ne terminent plus à $t = 1$ et lorsque les tâches C et D arrivent, les tâches C se retrouvent toutes ordonnancées et terminées (travail infinitésimal supplémentaire inclus) entre $t = 1$ et $t = 1 + O(\varepsilon \log n)$, car elles sont "poussées" hors de la file d'attente de **LAPS**$_{\frac{1}{2}}$ par les tâches B plus anciennes ; ces dernières ne terminerons finalement instantanément qu'une fois que les tâches D seront terminées, à $t = 2$. On constate donc que le temps de flot de tâches C a été *réduit* (d'un facteur $\varepsilon \log n$ arbitrairement petit) plutôt qu'allongé par l'ajout de travail parallèle (infinitésimal ou non d'ailleurs) ![54]

Il est très facile de générer ce même comportement à partir d'un ordonnancement de diffusion construit par **BLAPS**$_\beta$ pour tout $\beta > 0$. Ces phénomènes peuvent s'enchaîner en cascades et devenir incontrôlables. L'ajout de travail infinitésimal ne nous semble donc pas permettre de comparer les temps de flot des tâches augmentées, avec ceux des fichiers correspondants dans l'algorithme **BLAPS**. L'argumentation proposée par [97] pose donc problème. Il convient d'être extrêmement prudent en manipulant des infiniment petits. Il nous a paru que l'introduction de phases **ZOMB** et la démonstration de la compétitivité de **LAPS**$_\beta$ en présence de phase **ZOMB** soient le seul moyen rigoureux d'obtenir l'effet recherché, et donc que le théorème 3.32 soit la bonne façon de présenter le théorème 3 de [97].

7 Conclusions et perspectives

Avec Julien Robert, nous avons démontré que l'introduction dépendances arbitraires dans le modèle très général d'Edmonds [93] ne modifie pas la charge maximale supportée par un serveur pour peu qu'il partage les ressources entre les tâches avant de les partager entre leurs processus. Deux contributions non-négligeables de nos travaux sont la clarification des résultats d'Edmonds et Pruhs et d'avoir démontré qu'il était possible d'étendre encore le modèle. Nous avons également proposé un premier algorithme non clairvoyant compétitif pour gérer le cas de requêtes portant sur des ensembles de fichiers auprès d'un serveur *multicast*.

Une suite logique à nos travaux est d'étudier la compétitivité des algorithmes non clairvoyants avec des dépendances et des tâches plus complexes pour le cas du serveur *multicast*, en introduisant des contraintes de précédence et des phases arbitraires. On pourrait p. ex. imaginer deux requêtes, une pour deux tâches, A puis B, et une autre pour C puis le même B, où une seule exécution de B suffirait pour satisfaire les deux requêtes, pour peu que B soit exécutée après A et C. Il s'agirait de satisfaire p. ex. des clients qui désirent écouter leurs émissions de radio dans un certain ordre. Cela pourrait également capturer des situations plus complexes où certaines parties de projets disjoints peuvent être fusionnées. Un exemple très actuel d'une telle situation (cf. l'utilisation croissante des cartes graphiques pour mener des calculs intensifs) est le cas de cartes multi-processeurs SIMP *(Single Instruction Multiple Processors)* sur lesquelles on peut exécuter le même code sur des données différentes simultanément. On peut alors envisager d'exécuter des parties identiques d'applications différentes, chacune sur leurs données propres, simultanément sur la même carte sans surcoût (ce genre

[54] Notons le temps de flot *total* de l'ordonnancement (b) est supérieur à celui de (a). Il est donc possible que malgré tout, le temps de flot de **LAPS**$_\beta$ soit quant-à-lui monotone. Cette propriété mériterait d'être étudiée plus en avant. Remarquons cependant que la monotonie du temps de flot de **LAPS**$_\beta$ ne résoudrait en rien le problème concernant le temps de vie des processus que l'on souhaite égal au temps de vie qu'on leur a déclaré dans **BLAPS**$_\beta$.

d'opération est facilité par le recours massif aux librairies pour sous-traiter certaines parties des programmes). Ce genre d'application est tout à fait envisageable sur un serveur de calcul dédié à un certain type de simulations comme les prévisions météorologiques. Un autre exemple d'application serait la préparation d'un journal télévisé où à intervalle régulier (tous les soirs à 20h) toutes les informations disponibles (c.-à-d., dont la préparation est achevée) sont diffusées toutes ensemble dans une tâche unique. Les premières explorations que nous avons menées dans cette direction semblent indiquer que les performances ne dépendront plus seulement de la largeur κ de la structure de dépendance, mais aussi de sa profondeur, et cela même si toutes les tâches sont parallèles.

Nous l'avons évoqué précédemment, la réduction de la diffusion de fichiers à l'ordonnancement de tâches (cf. section 6) a atteint ses limites avec la preuve de la compétivité de **LAPS**$_\beta$ pour une charge arbitrairement proche de 100%. Il faudra procéder différemment pour obtenir des algorithmes de diffusion de fichiers qui soient compétitifs pour une charge supérieure à 50%, puisqu'un facteur 2 sur la charge est perdu lors de la réduction. Dans un article à paraître, Im et Moseley ont apparemment réussi à proposer tout récemment une nouvelle famille d'algorithmes non clairvoyants **LA-W**$_\varepsilon$ qui soit $(1+\varepsilon)$-rapide $O(1)$-compétitive pour la diffusion de fichiers de taille unitaire sans dépendance, pour tout $\varepsilon > 0$ [166]. Leur algorithme est basé sur la remarque suivante. L'algorithme **LWF**, qui diffuse systématiquement le fichier pour lequel l'attente cumulée est maximale, présente la faiblesse suivante : il prend en compte bien trop tard le cas d'un très grand nombre de requêtes demandant toutes à peu près simultanément le même fichier ; ces groupes de requêtes très serrées accumulent du coût très rapidement et finissent par coûter bien trop cher à **LWF** par rapport à l'optimum ; la démonstration de [98] que **LWF** n'est pas s-rapide $O(1)$-compétitif pour tout $s < \frac{1+\sqrt{5}}{2}$, exploite justement cette faiblesse. [166] propose alors de modifier la stratégie de **LWF** pour prendre également en compte les dates d'émission des requêtes en résolvant une optimisation multi-critères. Il sera très intéressant de voir si notre travail sur les dépendances peut s'adapter à leur étude et de déterminer s'il existe aussi un algorithme non clairvoyant $(1+\varepsilon)$-rapide $O(1)$-compétitif pour le *multicast* avec dépendances.

Concernant l'ordonnancement de tâches, le résultat d'Edmonds et Pruhs [99] sur la compétitivité de **LAPS**$_\beta$ pour toute vistesse $s > 1+\beta+\varepsilon$ ne résout pas encore tout à fait la question, puisque chaque algorithme **LAPS**$_\beta$ a une charge limite $\sim 1-\beta$ au-delà de laquelle il n'est plus $O(1)$-compétitif. Edmonds conjecture qu'il n'existe pas d'algorithme qui soit s-rapide $O_s(1)$-compétitif pour tout $s > 1$. Il propose même l'esquisse d'une preuve dans [92] utilisant des techniques très originales, mais ce travail n'est pas encore assez abouti pour clore définitivement la question. C'est en tout cas une piste que nous aimerions explorer et clarifier.

Une autre question est celle du coefficient d'étalement, le paramètre que nous avons introduit pour démontrer la compétitivité de **LAPS**$_\beta \circ$ **EQUI** pour l'ordonnancement de tâches avec contraintes de précédence. Actuellement, la valeur de ce coefficient dépend de l'algorithme d'ordonnancement de processus choisi et il n'est donc pas automatique de pouvoir convertir un minorant de ce coefficient en un minorant du meilleur facteur de compétitivité possible pour tout algorithme non clairvoyant. Julien Robert explore dans sa thèse une piste prometteuse, celle des *tâches remplisseuses*, sorte de tâches paramétrées que l'on peut injecter dans n'importe quel ordonnancement engendré par un algorithme non clairvoyant pour y provoquer des chutes de performances [263]. La structure de ces tâches s'inspire des pires instances obtenues en maximisant le coefficient d'étalement, mais la conversion n'est pas directe. Peut-être que cette conversion serait facilitée si l'on arrivait à définir ce coefficient indépendamment de l'algorithme utilisé. On obtiendrait alors un paramètre de la structure d'une tâche qui me-

surerait précisément l'impact maximal de la structure de la tâche sur les performances de tout algorithme. Nous pouvons également regretter que la maximisation du coefficient d'étalement utilise massivement les phases **SEQ**. Ces phases apparaissent naturellement via la réduction **SEQ-PAR** de la proposition 3.2, même lorsque toutes les tâches sont **PAR** au départ. Mais il n'est pas clair que l'inverse soit vrai : rien ne garantit que les valeurs maximales du coefficient d'étalement obtenues sur les instances **SEQ-PAR**, puissent également être obtenues avec des instances dont les tâches traversent uniquement des phases **PAR**.

4 Auto-assemblage de formes en temps réel

Ce chapitre présente les travaux [39] que nous avons réalisés avec Éric Rémila et Florent Becker pendant leurs deux séjours au Chili en 2006 et 2007.[55] Ces travaux portent sur l'auto-assemblage. Le principe est de définir des petites tuiles de telle façon qu'elles se collent naturellement les unes aux autres par des interactions locales, pour faire émerger la forme désirée. Il s'agit d'une approche différente de l'émergence étudiée au chapitre 1. Nous cherchions alors à comprendre les interactions locales et globales qui permettent d'*expliquer* l'émergence d'un phénomène global *préexistant*, le phénomène petit-monde. Nous envisageons maintenant l'émergence sous l'angle inverse. Nous avons ici la maîtrise totale des règles du jeu et nous cherchons à forcer l'émergence d'un phénomène global. En ce sens, cette situation ressemble à celle du chapitre 3 où nous souhaitions voir émerger un régime permanent garant de la non-divergence du système. Mais encore une fois, la situation était différente puisque nous ne connaissions rien de ce que nous faisions ni de ce que nous devions faire, et cherchions malgré tout à obtenir et démontrer que des stratégies performantes peuvent y parvenir. L'émergence peut donc s'étudier sous des angles très variés.

Nous allons voir que les règles imposées par l'auto-assemblage d'une forme contraignent très fortement la géométrie de la construction. Nous adoptons une fois de plus le point de vue d'un informaticien en étudiant l'optimisation du temps de construction des formes. Nous allons démontrer que la minimisation du temps impose des contraintes fortes sur les flux d'information circulant dans la forme en construction, et que l'on peut en déduire un jeu de tuiles original assemblant ces formes en temps optimal. Le jeu de tuiles obtenu présente une caractéristique inattendue : la construction est structurée par un ensemble de trois signaux *interdépendants* où *aucun* n'est maître l'un de l'autre et où chacun dépend des deux autres pour sa progression, un phénomène à notre connaissance absent des domaines pourtant proches des automates cellulaires et des pavages.

[55] Bizarrement, il est souvent plus simple de collaborer avec des collègues lorsque l'on habite à plusieurs milliers de kilomètres plutôt qu'à quelques bureaux l'un de l'autre.

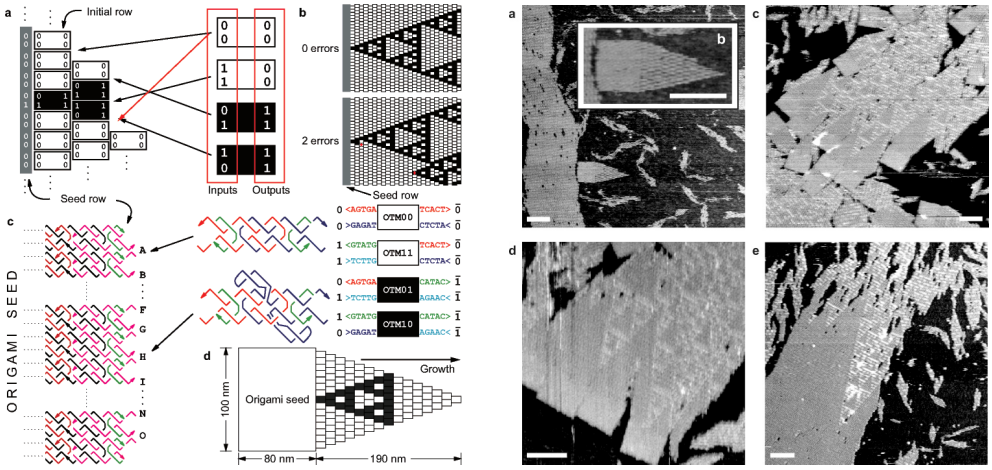


Figure 4.1 – Images prodigieuses, extraites de [125], de la réalisation à une échelle nanométrique de l'automate cellulaire des triangles de Sierpinski par auto-assemblage de tuiles construites séparément en repliant astucieusement des brins d'ADN. *(le procédé d'imagerie par microscopie à force atomique est responsable en partie du déchirement des motifs)*

1 Introduction

La découverte des propriétés étonnantes des nano-particules [301] et les possibilités ouvertes par la construction de nano-robots, font de leur réalisation à grande échelle un enjeu majeur actuellement. Or, seul un petit nombre de structures nanoscopiques peuvent être obtenues directement par réactions chimiques. Les premières approches pour obtenir une forme particulière ont proposé de la construire atome par atome au microscope à effet tunnel, méthode initiée par Eigler et Schweizer pour IBM en 1990 [101]. La construction est alors très lente et ne permet de construire que des objets relativement simples et en petit nombre. L'auto-assemblage est alors apparu comme une alternative intéressante. Le principe est de décomposer l'objet que l'on souhaite construire en petits morceaux, plus faciles à réaliser en nombre et de faire en sorte que les morceaux s'agglutinent naturellement les uns aux autres dans les bonnes positions pour former l'objet désiré. Les cristaux et les récifs coralliens sont des exemples de processus naturels de croissance par agrégation.

Winfree propose dans sa thèse [321] un cadre algorithmique pour reproduire ces constructions de façon artificielle. Les travaux de Rothemund sous sa direction [275, 272] puis leur amélioration dans [125] (v. illustration fig. 4.1) ont démontré que ce cadre est effectivement réalisable en pratique : la figure 4.1 (à droite) présente la simulation d'un calcul des triangles de Sierpinski par automate cellulaire réalisée dans un bécher selon le principe imaginé par Winfree. Ce principe est simple (au moins sur le papier) [321, 322, 296, 275]. On fabrique des brins d'ADN qui se replient sur eux-mêmes pour former des briques carrées (fig. 4.1 à gauche). Ces brins sont ensuite dupliqués à volonté par réaction de polymérisation en chaînes, un procédé à croissance exponentielle maintenant classique et totalement automatisé en biologie. Chaque brin est choisi pour que chaque côté de la brique ne puisse se lier qu'avec un bord de même type et avec une certaine force (de valeur 1 ou 2). Une fois les briques construites, on les plonge dans une solution à 90°C (on parle de *one-pot reaction*) et on attend qu'elles s'assemblent toutes seules en laissant refroidir la solution à température ambiante. Ce procédé expérimental assure que seules les briques qui sont attachées à un agrégat par au moins

un côté de force 2 ou deux côtés de force 1, resteront attachées jusqu'à la fin du processus. La formation de l'agrégat final se fait donc par ajout successif de briques qui s'attachent à l'agrégat courant par un côté de force 2 ou deux côtés de force 1. On peut admirer le résultat sur la figure 4.1 (à gauche). Outre sa réalisabilité en pratique, ce modèle est Turing-complet et permet donc de développer une algorithmique propre pour construire les formes que l'on souhaite. C'est ce deuxième aspect qui va nous intéresser ici.

Spécificités algorithmiques de l'auto-assemblage. La conception d'algorithmes d'auto-assemblage pose plusieurs difficultés. tout d'abord, il faut pouvoir contrôler l'ordre de l'assemblage au moins localement afin d'éviter un non-déterminisme dans la structure de l'objet final. Ensuite, comme l'assemblage se fait par ajouts successifs, la première tuile posée sur chaque colonne ou chaque ligne est nécessairement attachée à l'agrégat courant par côté de colle de force 2 et les suivantes sur cette colonne ou cette ligne, le seront nécessairement par deux côtés de force 1. Ceci impose des contraintes sur le flux d'information traversant la forme que l'on veut construire, et cela impose par exemple que, contrairement à l'algorithmique classique des automates cellulaires (le *firing squad* par exemple), les briques de remplissage véhiculent de l'information depuis la brique initiant la colonne ou la ligne.

Le modèle de Winfree. Le cadre défini par Winfree consiste en un raffinement des tuiles de Wang où les bords n'ont plus seulement une couleur, mais également une *force* qui représente l'intensité de l'affinité entre deux bords de la même couleur. On parle alors de *colle*. Comme pour les tuiles de Wang, seuls deux bords ayant des colles de la même couleur peuvent s'accrocher l'un à l'autre. Le processus d'auto-assemblage est contrôlé par un entier τ appelé *température*. Une tuile reste attachée à l'agrégat si et seulement si la somme des forces de ses attaches avec l'agrégat vaut au moins τ. Sinon, la tuile se détache de l'agrégat sous l'effet de l'agitation thermique. La construction de l'agrégat est démarrée par une tuile spéciale, appelée l'*amorce*.

Travaux théoriques antérieurs. La plupart des travaux précédents cherchent à minimiser le nombre de tuiles utilisées pour construire une figure fixée (un paramètre très important pour la faisabilité en pratique de l'algorithme). Ce modèle permet d'assembler à température $\tau = 2$ n'importe quelle forme bi-dimensionnelle en utilisant $O(K/\log K)$ tuiles, où K est sa complexité de Kolmogorov [9, 296]. La principale préoccupation est alors de décoder la représentation compacte de la forme, plutôt que la rapidité de la construction. Par exemple, [38] obtient des jeux ayant le nombre minimum de tuiles pour assembler un certain nombre de formes. La figure 4.2 présente leur construction du carré $n \times n$ avec nombre minimum de tuiles (cinq). La construction démarre de la tuile étoilée du coin en bas à gauche et procède en construisant la diagonale qui s'étend ensuite vers les bords pour remplir le carré à l'aide de tuiles de remplissage (roses au dessus et turquoises en dessous). La progression de la diagonale s'arrête lorsqu'une tuile de remplissage turquoise se fixe sur la diagonale à la place de la tuile étoilée. La réutilisation de cette tuile de remplissage pour provoquer l'arrêt de la construction est l'astuce qui permet d'obtenir un jeu de tuiles optimal. [38] démontre en effet qu'il est impossible de ne construire que des carrés si l'on ne dispose que de quatre tuiles.

Construction d'un jeu de tuiles par contraintes temporelles. À notre connaissance, seuls des résultats asymptotiques sont connus sur le temps de construction. Une question récurrente naturelle posée aux auteurs de [38] à chaque présentation de leur travaux sur le nombre minimal de tuiles requis pour construire certaines figures était : mais cette construction est-elle optimale ? Peut-on faire mieux ? C'est cette question que nous étudions ici. Nous

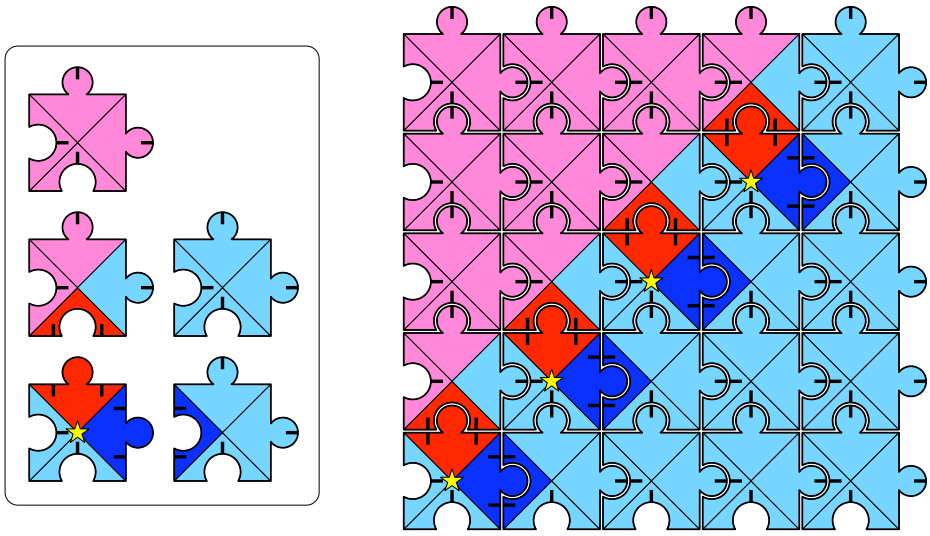


Figure 4.2 – Construction du carré $n \times n$ avec un jeu de tuiles minimal (cinq tuiles) [38].

nous proposons d'étudier les constructions en temps *optimal* et d'utiliser cette contrainte pour construire le jeu de tuiles. À partir de cette contrainte d'optimalité, nous reconstruisons la façon dont l'information doit circuler dans la figure d'où nous déduisons un ordre optimal d'assemblage, duquel nous déduirons le jeu de tuiles. Cette approche *top-bottom* où l'on s'assure d'abord des propriétés avant de construire un jeu de tuiles les vérifiant permet de simplifier considérablement les preuves par rapport à l'approche inverse où il faut envisager toutes les associations possibles de tuiles. Un fait étonnant est que contrairement à la plupart des constructions jusqu'à nos travaux, comme celle de [38] dirigée par le signal de la diagonale (v. fig. 4.2), nous obtenons une construction en temps optimal des carrés sans utiliser de signaux "maître" (v. fig. 4.5). La progression de chacun des trois signaux dépend de celles des autres. À notre connaissance ce type d'interdépendance entre signaux n'était connue dans aucun des trois domaines directement reliés à ce modèle : les automates cellulaires, les pavages, ou l'algorithmique distribuée. La notion de signal est donc peut-être significativement différente en auto-assemblage.

Afin de définir proprement la notion de temps associé à un jeu de tuiles, ce jeu de tuiles devra pouvoir construire des formes de taille arbitrairement grande. Nous étudierons donc des jeux de tuiles pouvant assembler n'importe quelle taille de la forme désirée, la taille se décidant de façon non-déterministe. Nous verrons qu'il est toujours possible d'en déduire un jeu pour une taille précise, en lui superposant un automate qui décode la taille et arrête la construction au bon moment, comme dans [38].

Notre contribution. Dans ce chapitre, nous étudions le cas des carrés et des cubes et démontrons qu'il est possible de les auto-assembler en *temps réel* (c.-à-d. au plus tôt). Nous obtenons un jeu de tuiles qui assemblent les carrés de côté n en temps $2n - 2$, ce qui améliore le temps $3n - 5$ de [38].

La description d'un jeu de tuiles peut se révéler très délicate en 3D et la preuve de sa correction plus encore, car les signaux ne partitionnent plus l'espace. Nous verrons alors tout l'intérêt de notre méthode où notre jeu de tuiles est déduit de l'*ordre d'assemblage* des tuiles après la vérification de quelques conditions de régularité et de déterminisme local. En

étendant l'analyse 2D à la 3D, nous obtenons ainsi une présentation intelligible à la fois de la description des tuiles et de la preuve de leur correction. Notre preuve garantit en particulier l'absence de *bulles* lors la construction.

2 Jeux de tuiles auto-assemblants

Nous donnons dans cette section les définitions formelles du modèle que nous avons décrit dans l'introduction.

2.1 Le modèle

Définition 4.1 (Tuiles). *Une* tuile *est un carré unitaire ayant une* colle *sur chaque côté. On la représente par le quadruplet de ses colles $\langle \alpha_N, \alpha_S, \alpha_E, \alpha_W \rangle$ choisies dans un ensemble fini Σ.*[56] *Chaque colle $\alpha \in \Sigma$ a une* force $g(\alpha) \in \mathbb{N}$. *Deux tuiles ne peuvent s'attacher que si leurs bords communs sont de la même colle α, on dit alors qu'elles sont jointes par un* lien *de force $g(\alpha)$.*

On représentera dans les figures la force de chaque colle par autant de petits traits perpendiculaires au côté où elle est appliquée. Sur la figure 4.2, les colles rouge et bleue sont de forces 2 tandis que les colles rose et turquoise sont de force 1 et qu'aucune colle n'est de force 0 ou > 2.

Définition 4.2 (Jeu de tuiles). *Un* jeu de tuiles à amorce $\mathcal{T}$ *est la donnée de :*

- $T \subseteq \Sigma^4$, *l'ensemble de ses tuiles dont les colles appartiennent à un alphabet fini Σ.*
- $t_0 \in T$, *une tuile particulière appelée l'*amorce.
- $\tau \in \mathbb{N}^*$, *sa température nominale.*

Les "formes" construites par un jeu de tuiles sont celles obtenues par agrégation depuis la tuile-amorce, suivant la règle qu'une tuile ne peut s'attacher à l'agrégat que si la somme des forces de ses liens avec lui est supérieure ou égale à la température.

Définition 4.3 (Productions). *Une* configuration *est une fonction partielle $A : \mathbb{Z}^2 \to \mathcal{T}$, telle que deux tuiles voisines ont la même colle sur leur côté commun. Le domaine de A,* dom A, *est appelé la* forme réalisée par A. *Nous dirons qu'une configuration B* étend *une configuration A (noté $A \vdash_T B$) sur un site (i, j) si les quatre conditions suivantes sont vérifiées : (a) $(i, j) \notin \operatorname{dom} A$; (b) $\operatorname{dom} B = \{(i, j)\} \cup \operatorname{dom} A$; (c) $B_{|\operatorname{dom} A} = A$; (d) la tuile placée dans B sur le site (i, j) est arrimée à ses voisines dans A, c.-à-d. la somme des forces de ses liens avec elles vaut au moins τ. On désigne par $\vdash_{\mathcal{T}}^*$ la fermeture transitive de $\vdash_{\mathcal{T}}$.*

Les productions $\mathcal{P}$ *d'un jeu de tuiles $\mathcal{T}$ sont les configurations dérivées de la* configuration initiale Γ_0, *ayant pour seule tuile, la tuile-amorce t_0 en $(0, 0)$: $\mathcal{P} = \{A : \Gamma_0 \vdash_{\mathcal{T}}^* A\}$. Nous dirons qu'une production est* finale *si aucune autre production ne peut en être dérivée. L'ordre induit par $\vdash_{\mathcal{T}}^*$ depuis la configuration initiale est appelée la* dynamique *de $\mathcal{T}$.*

On représentera le plus souvent la tuile-armorce avec une étoile dans les figures. Dans le cas de la figure 4.2 où la tuile-amorce apparaît plusieurs fois dans les productions, la tuile-amorce de la configuration initiale est celle située dans le coin en bas à gauche du carré.

[56] Remarquons que l'on impose l'orientation des tuiles. Il est toujours possible de la forcer en codant l'orientation dans les colles.

2.2 Jeux de tuiles ordonnés

Une propriété souhaitable d'un jeu de tuile est que l'ordre (partiel) d'assemblage d'une production donnée soit fixé entre tuiles voisines. Cette propriété est en effet très commode pour valider le comportement [274]. Une condition classique est la *condition RC* [274] selon laquelle les tuiles ne peuvent être placées sur une ligne (resp. colonne) qu'après qu'une tuile déterminée à l'avance ait été placée sur cette ligne (resp. colonne). On dit alors que cette tuile particulière *ouvre* la ligne (resp. colonne). Mais cette condition est un peu trop restrictive et interdit p. ex. la construction de forme concave. Aussi nous lui préférerons la condition suivante plus faible qui donne plus de liberté tout en conservant les garanties nécessaires de déterminisme pour la conception et l'analyse des jeux de tuiles.

Définition 4.4 (Condition d'ordre). *Pour tout $\gamma = \langle \Gamma_0 \vdash_{\mathcal{T}} A_1 \vdash_{\mathcal{T}} A_2 \vdash_{\mathcal{T}} \cdots \vdash_{\mathcal{T}} A_t = P \rangle$ assemblage d'une production P et deux sites voisins (i,j) et (i',j') de P, nous dirons que $(i,j) \prec_{P,\gamma} (i',j')$ si (i,j) a été couvert par une tuile avant (i',j') dans l'assemblage γ.*

Nous disons qu'un jeu de tuiles $\mathcal{T}$ est ordonné *si pour toute production P, la relation $\prec_{P,\gamma}$ entre les sites P est indépendante de γ. Dans ce cas, cette relation $\prec_P$ définit un ordre partiel (v. [76]) appelé* ordre local *des sites de P.*

L'assemblage de toute production P par un jeu de tuiles ordonné est déterministe "localement" : deux tuiles voisines sont toujours placées dans le même ordre. La seule incertitude réside dans le choix non-déterministe de la tuile qui sera placée sur chaque site. Ce choix conditionne la production qui sera obtenue. Quelques propriétés élémentaires des jeux de tuiles ordonnées sont :

- Si une production Q étend une production P, l'ordre local $\prec_Q$ coïncide avec $\prec_P$ sur les sites de P. De plus, puisqu'aucun site ne peut être couvert avant ses prédécesseurs, P est un idéal[57] de $\prec_Q$.
- Puisque qu'avoir plus de τ prédécesseurs induirait du non-déterminisme dans la relation de précédence d'un site par rapport à ses voisins, chaque site d'une production P a au plus τ prédécesseurs dans $\prec_P$ et l'amorce t_0 est la seule tuile sans prédécesseur.
- Tout jeu de tuile RC [274] est ordonné.

2.3 Particularités de l'auto-assemblage

Comparaison avec des modèles classiques. Il est tentant de s'inspirer des algorithmes développés pour les automates cellulaires. Il faut cependant tenir compte des différences importantes suivantes :

- Le choix d'une tuile est *irrévocable* : un site ne peut donc pas "changer d'état", ce qui contraint considérablement les déplacements des "signaux" qui ne peuvent par exemple, pas "rebondir".
- *Tout* ce qui peut s'attacher, s'attachera : il faut donc se garder des risques d'"auto-allumage" des signaux. Par exemple, les tuiles de remplissages au dessus et en dessous de la diagonale de la construction du carré figure 4.2 ont des colles de couleurs différentes pour cette raison : si elles utilisaient la même colle, une nouvelle diagonale

[57] Un *ideal* P d'un ordre partiel $\prec$ est un ensemble tel que pour tout site $(i,j) \in P$, si $(i',j') \prec (i,j)$ alors $(i',j') \in P$ (v. [76]).

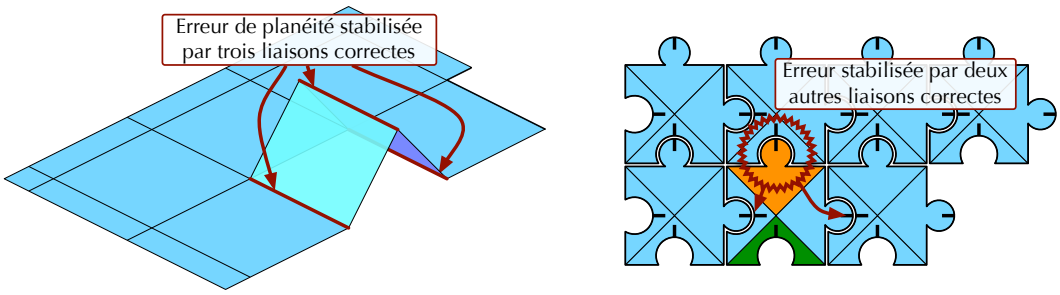


Figure 4.3 – Exemple d'erreurs expérimentales en auto-assemblage.

pourrait démarrer à tout instant au-dessus ou en-dessous de la diagonale légitime, ruinant la construction.

Le moyen de contourner ces difficultés est d'imposer la condition d'ordre que nous avons vue précédemment. Cette condition a alors les conséquences suivantes :

- Pour toute production P, il n'existe qu'un seul "signal principal" par coordonnée, donné par la première tuile posée sur chaque ligne ou colonne (par composante connexe de l'intersection de P avec cette ligne ou cette colonne). Cette tuile est aussi nécessairement la seule fixée avec une colle de force $\geqslant \tau$.
- Ainsi, les tuiles de remplissages doivent véhiculer l'information depuis le "signal principal" pour permettre l'existence d'autres signaux : il n'existe pas de tuile quiescente passive en auto-assemblage.
- Il existe donc des flux d'information à l'intérieur de toute production : depuis le "signal principal" vers les bouts de chaque ligne et colonne.

Toutes ces remarques font qu'il existe des différences significatives entre l'algorithmique de l'auto-assemblage et celle des automates cellulaires.

Difficultés expérimentales. Il existe, on s'en doute, de nombreux obstacles à la réalisation dans un becher des algorithmes d'auto-assemblage. Il est en effet courant que des tuiles s'attachent indûment à l'agrégat. Voici quelques sources d'erreurs possibles :

- Deux tuiles se fixent ensemble et se stabilisent l'une l'autre même si un de leur bord ne convient pas à l'agrégat (fig. 4.3, à droite).
- Erreurs de planéité : des tuiles se fixent ensemble en dehors du plan de la construction (fig. 4.3, à gauche).
- Fusion de deux agrégats : deux agrégats issues de deux tuiles-amorces différentes fusionnent.
- Auto-allumage : une construction démarre en l'absence de tuile-amorce.

La correction des erreurs a été depuis le départ une préocupation importante de l'équipe de Winfree. Il est possible dans certains cas d'apporter une réponse algorithmique, en superposant des codes correcteurs d'erreurs p. ex., ou en jouant sur l'ordre de construction, pour minimiser à tout moment l'exposition des "zones à risque". Il est aussi utile de jouer sur la rigidité des tuiles et la puissance des liaisons en modifiant leur structure chimique. On pourra se reporter à [66, 126] pour plus d'informations. Nous supposerons dans ce chapitre qu'aucune erreur ne se produit.

3 Squelette et temps réel

Nous ne considérerons maintenant que des jeux de tuiles ordonnés.

3.1 Jeux de tuiles en temps réel

Dynamique temporisée. L'assemblage d'une production est habituellement modélisé par une chaîne de Markov où chaque tuile tente de s'attacher en chaque site libre suivant un processus de Poisson de taux proportionnel à sa concentration [9]. Nous ne nous intéressons pas ici à la probabilité d'obtenir telle ou telle production mais à l'espérance de son temps d'assemblage conditionné par l'événement que l'on obtienne bien cette production. Nous savons par [9] que l'espérance du temps de construction d'une production P est proportionnelle à la longueur $\ell(P)$ de la plus longue chaîne de l'ordre $\prec_P$, où la constante multiplicative dépend uniquement des valeurs des concentrations des différentes tuiles. Comme nous conditionnons cette espérance par la probabilité d'atteindre une production donnée, le paramètre pertinent pour mesurer le temps de la construction en fonction de la taille n de la forme est $\ell(P)$. Les concentrations ne jouent en effet dans ce cas que le rôle des GHz pour les ordinateurs : les doubler diminue de moitié le temps de calcul. Nous dirons donc qu'un assemblage est *optimal en temps* si la longueur $\ell(P)$ de la plus longue chaîne de $\prec_P$ est la plus petite possible P.

Remarquons que $\ell(P)$ est exactement le temps de construction de la *dynamique parallèle*, où, à chaque pas de temps, toutes les tuiles qui peuvent s'arrimer sont systématiquement rattachées à l'agrégat courant. Notons également qu'aucune paire de tuiles voisines n'est placée simultanément dans la dynamique parallèle.

Comme tous les sites sont couverts les uns après les autres depuis la tuile-amorce, nous avons le minorant suivant sur la longueur de la plus longue chaîne d'une production P.

Fait 4.5 (Minorant trivial). *Pour tout forme 4-connexe $S \subset \mathbb{Z}^2$, tout jeu de tuiles $\mathcal{T}$ et toute production P réalisant S, nous avons : $\ell(P) \geqslant ||S||_1$, où $||S||_1$ désigne la distance-ℓ_1 maximum d'un site de S à la tuile-amorce placée en $(0,0)$.*

Définition 4.6 (Assemblage en temps réel). *Soit $\mathcal{S}$, une famille de formes. Nous disons qu'un jeu de tuiles $\mathcal{T}$ assemble la famille $\mathcal{S}$* en temps réel, *si $\mathcal{S}$ est l'ensemble exact des formes réalisées par les productions finales de $\mathcal{T}$, et si pour toute production* finale *P réalisant une forme $S \in \mathcal{S}$, nous avons : $\ell(P) = ||S||_1$.*

Le fait 4.5 démontre que tout jeu de tuiles temps-réel est optimal en temps.

Définition 4.7 (Rang). *Nous appelons* rang *$\rho_P(u)$ du site u d'une production P, la longueur de la plus longue chaîne de prédécesseurs menant de $(0,0)$ à u dans $\prec_P$. Nous dirons qu'un site u de P est* à l'heure *si $\rho_P(u) = ||u||_1$.*

Clairement, un jeu de tuile assemble une production P en temps réel si et seulement si les sites de rang maximum dans P sont à l'heure. Notons que $\rho_P(u)$ est la date exacte de couverture du site u de la production P dans la dynamique parallèle.

3.2 Rang et squelette d'une production

La définition suivante formalise la notion de "signal principal" dans les jeux de tuiles ordonnés que nous avons évoquée précédemment.

Définition 4.8 (Squelette). *Le* squelette-x *d'une production* P *(resp.,* squelette-y*) est l'union du site amorce avec les sites de* P *qui ont un unique prédécesseur dans* $\prec_P$*, placé à leur droite ou à leur gauche (resp., au dessus ou en dessous). Le* squelette *de* P *est l'union des deux squelettes-*x *et -*y*.*

Le squelette-x (resp. -y) d'une production P est l'ensemble des sites qui *ouvrent* l'axe des x (resp. des y) durant l'assemblage de P, c.-à-d. ceux qui sont les premiers à être couverts par une tuile dans chaque colonne (resp. ligne). En effet, comme le jeu de tuiles est ordonné, chaque colonne se remplit de proche en proche depuis le site du squelette-x de cette colonne. Remarquons que même si certaines tuiles de remplissage sont porteuses de signaux importants, elles seront couvertes par agrégation depuis et après les sites des squelettes-x et -y de la même colonne et de la même ligne respectivement. Nous pouvons donc relier le rang au squelette de la façon suivante. Le rang $\rho_P(u)$ d'un site u d'une production P est minoré par la plus grande de ces deux quantités :

$$\sigma_X(u) = \rho_P(a) + ||u - a||_1 \quad \text{et} \quad \sigma_Y(u) = \rho_P(b) + ||u - b||_1, \tag{4.1}$$

où a et b sont les sites les plus proches dans les squelettes-x et -y dans la colonne et la ligne de u dans P, respectivement (c.-à-d., où a et b sont les sites qui ont ouvert respectivement la colonne et la ligne de u durant l'assemblage de P).

Ces deux minorants suivent le "flux d'information" au sein d'une production depuis son squelette. Nous allons voir dans la section suivante que ces minorants vont nous permettre de *localiser* le squelette d'un jeu de tuiles temps-réel. Nous illustrerons cette approche en obtenant des jeux de tuiles temps-réel pour assembler les carrés puis les cubes enracinés en l'origine.

4 Assemblage des carrés en temps réel

Commençons par voir comment assembler en temps réel la famille des carrés dont le coin en bas à gauche est à l'origine, $S_n = \{0, \dots, n-1\}^2$ pour $n \geqslant 2$. Notre approche consiste à déterminer un ordre local entre les sites à partir des "flux d'information" imposés par les contraintes de temps réel dans le carré ; puis à en déduire un jeu de tuiles ordonné réalisant cet ordre.

4.1 Ordre local temps-réel pour les carrés

Comme $||S_{n+1}||_1 = 2n$, notre but est d'obtenir un ordre local des sites du carré S_{n+1} dont le rang maximum soit (au plus) $2n$.

Considérons un jeu de tuiles arbitraire $\mathcal{T}$ réalisant le carré S_{n+1}. Comme S_{n+1} est convexe et $\mathcal{T}$ ordonné, chaque colonne (resp., ligne) de S_{n+1} contient exactement un site du squelette-x (resp., -y). En effet, si deux sites d'une même colonne appartenaient au squelette-x, les tuiles situées entre eux (eux compris) pourraient être arrimées par au dessous ou par en dessous et l'ordre local d'assemblage dépendrait de la séquence d'assemblage choisie : $\mathcal{T}$ ne serait donc pas ordonné.

Désignons donc par $a_i = (i, y_i)$ et $b_j = (x_j, j)$ les positions des squelettes-x et -y dans les différentes colonnes et lignes respectivement. Notons que $y_n = n - 1$ dans les constructions de [38, 273]. Maintenant, pour que l'assemblage ait lieu en temps réel, il faut que le rang

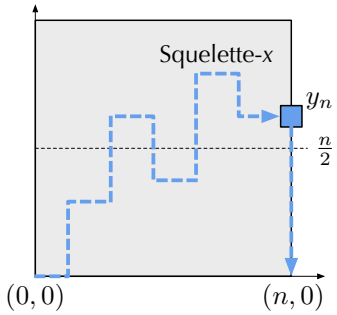


Figure 4.4 – Le squelette-x ne peut pas monter trop haut dans un assemblage temps-réel.

du coin en bas à droite $(n,0)$ soit au plus $2n$. Ainsi, par la minoration (4.1) (v. illustration fig. 4.4) :

$$2n \geqslant \rho_{S_{n+1}}(n,0) \geqslant \rho_{S_{n+1}}(a_n) + ||(n,0)-(n,y_n)||_1 \geqslant ||a_n||_1 + y_n = n + 2y_n.$$

Nous en concluons que pour tout jeu de tuiles temps-réel ordonné de S_{n+1}, $y_n \leqslant \frac{n}{2}$ et que les squelettes-x et -y ne peuvent donc monter au-dessus du site $(n, \lfloor n/2 \rfloor)$ et à la droite du site $(\lfloor n/2 \rfloor, n)$ respectivement.

Nous considérons donc les squelettes-x et -y suivants (v. fig.4.5 à gauche) :

$$a_i = (i, \lfloor \tfrac{i}{2} \rfloor) \quad \text{et} \quad b_j = (\lfloor \tfrac{j}{2} \rfloor, j).$$

La minoration (4.1) donne que le rang de tout site $u = (i,j)$ dans tout ordre local compatible avec ce squelette vaut au moins : $\sigma(u) = \max(||a_i||_1 + ||u - a_i||_1, ||b_j||_1 + ||u - b_j||_1)$. Le calcul de $\sigma(u)$ donne que :

- $\sigma(u) = i + j = ||u||_1$ si $j \geqslant \lfloor \frac{i}{2} \rfloor$ et $i \geqslant \lfloor \frac{j}{2} \rfloor$, c.-à-d., si u est situé entre les deux squelettes (la zone verte de la figure 4.5 à gauche) ;
- $\sigma(u) = i + 2\lfloor \frac{i}{2} \rfloor - j > ||u||_1$ si $j < \lfloor \frac{i}{2} \rfloor$, c.-à-d., si u est situé en dessous du squelette-x (la zone violette) ;
- $\sigma(u) = j + 2\lfloor \frac{j}{2} \rfloor - i > ||u||_1$ si $i < \lfloor \frac{j}{2} \rfloor$, c.-à-d., si u est situé au dessus du squelette-y (la zone rose).

Clairement, $\sigma(u) \leqslant 2n$, pour tout site u. On peut alors vérifier facilement que σ est le rang de l'ordre local suivant, où les prédécesseurs de chaque site $u = (i,j)$ sont (v. fig. 4.5 à gauche) :

- son voisin de gauche $(i-1, j)$, si $u = a_i$ appartient au squelette-x *(en bleu)* ;
- son voisin du dessous $(i, j-1)$, si $u = b_j$ appartient au squelette-y *(en orange)* ;
- ses voisins de gauche et du dessus, $(i-1, j)$ et $(i, j+1)$, si u est situé en dessous du squelette-x *(en violet)* ;
- ses voisins de gauche et du dessous, $(i-1, j)$ et $(i, j-1)$, si u est situé entre les deux parties du squelette *(en vert)* ;
- ses voisins de droite et du dessous, $(i+1, j)$ et $(i, j-1)$, si u est situé au dessus du squelette-y *(en rose)* ;

Les flèches en noires de chaque case de la figure 4.5 (à gauche) indiquent les prédécesseurs de chaque tuile pour cet ordre. Il ne nous reste plus qu'à implémenter cet ordre sous la forme d'un jeu de tuiles ordonné, pour obtenir *directement* une construction du carré par auto-assemblage en temps-réel.

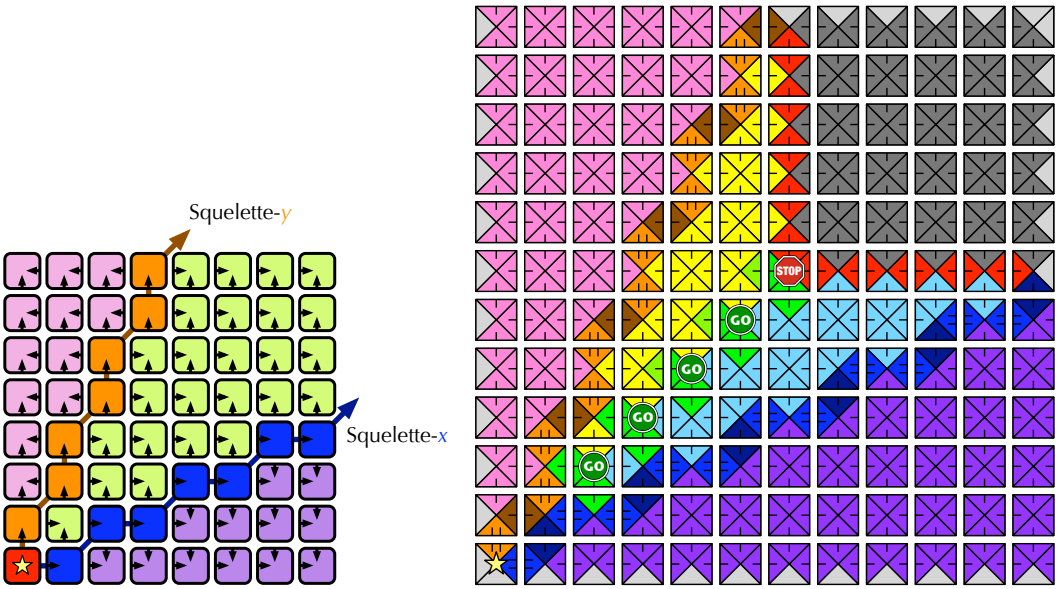


Figure 4.5 – Le squelette, son ordre induit et un jeu de tuiles assemblant le carré en temps réel.

4.2 Jeu de tuiles temps-réel pour carrés

La difficulté principale pour implémenter l'ordre temps-réel défini ci-dessus, est la *synchronisation* des deux signaux *distants* portant les squelettes-x et -y.

L'astuce consiste à remarquer que toutes les tuiles situées entre les deux parties du squelette sont toutes placées *à l'heure* (leur rang est égal à leur norme-ℓ_1). Nous pouvons alors utiliser les tuiles de cette zone pour porter le signal de synchronisation juste à temps pour informer simultanément les deux parties du squelette.

La figure 4.5 (à droite) illustre notre procédé dans le cas où le carré construit est de côté pair. La progression des signaux portant les squelettes-x et -y est contrôlée par le choix de la tuile placée sur la diagonale :

- Si une tuile `GO` est placée sur la diagonale, le signal `GO` se propage le long de la ligne *(en turquoise)* et de la colonne *(en jaune)*, et arrive *juste à temps* pour décaler les signaux portant les squelette-x et -y d'un cran vers le haut et vers la droite respectivement, et les autoriser à progresser de deux pas en autonome.
- Si une tuile `STOP` est placée sur la diagonale, le signal `STOP` se propage le long de la ligne et de la colonne *(en rouge)* et arrête *juste à temps et simultanément* la progression des squelettes-x et -y. Les tuiles de remplissage comblent les sites encore inoccupés et terminent la construction du carré. Afin de pouvoir construire des carrés de côté pair et impair, nous avons deux types `pair` et `impair` de tuiles `STOP`. La tuile `STOP-impair` ordonne aux squelettes de progresser d'un pas supplémentaire avant de terminer.

La construction est clairement divisée en cinq zones *(rose, jaune, grise, turquoise et violette sur la figure 4.5 à droite)*. Nous pouvons alors en déduire très facilement le jeu de tuile en considérant uniquement les relations de précédence définies par l'ordre induit par le squelette, et en prenant garde d'utiliser des colles différentes dans chaque zone.[58]

[58] Il est également possible d'utiliser les méthodes à base de signaux développer par Florent Becker durant sa

Théorème 4.9 ([39]). *Il existe un jeu de tuiles auto-assemblantes, ordonné et temps-réel dont les productions finales à température 2 sont exactement les carrés S_n.*

Nous ne donnerons pas les détails de la preuve ici qui consiste *uniquement* à vérifier par induction que chaque tuile s'insère au bon endroit dans l'ordre défini par l'ordre induit par les squelette-x et -y (fig. 4.5 à gauche).

Remarquons que le jeu de tuiles présenté à la figure 4.5 (à droite) n'est pas minimal et que l'on peut réduire significativement le nombre de tuiles en décalant les squelettes-x et -y tout en conservant l'optimalité en temps. Il est également facile de contrôler la taille du carré à l'aide d'un compteur comme [273], placé sur les tuiles de la zone *à l'heure* et forçant l'apparition à la coordonnée $(\lfloor \frac{n}{2} \rfloor, \lfloor \frac{n}{2} \rfloor)$ de la tuile `STOP pair` ou `impair` suivant la parité de n. L'ajout de $O(\log n / \log\log n)$ tuiles supplémentaires suffit. Cette méthode peut être utilisée pour lancer quatre constructions de carrés identiques vers le SE, SW, NW et NE, et obtenir ainsi une construction en temps optimal d'un carré enraciné en un site quelconque.

5 Assemblage des cubes en temps réel

Notre démarche permet d'obtenir avec un minimum d'effort un jeu de tuiles 3D assemblant en temps réel la famille des cubes $C_n = \{0, \dots, n-1\}^3$ pour $n \geqslant 2$. Il s'agit à notre connaissance de la première construction 3D par auto-assemblage. Nous étendons canoniquement les définitions des sections 2 et 3 de la 2D à la 3D.

5.1 Le squelette

Comme nous l'avons observé à la section 3.2, l'assemblage ordonné d'une forme convexe implique l'existence d'un squelette ayant une unique branche par dimension : l'ensemble des sites ouvrant chaque dimension. Nous nous inspirons directement du cas des carrés, pour définir les trois squelettes-x, -y et -z, reliant l'origine $(0,0,0)$ aux centres des trois faces opposées du cube :

$$a_i = (i, \lfloor \tfrac{i}{2} \rfloor, \lfloor \tfrac{i}{2} \rfloor), \quad b_j = (\lfloor \tfrac{j}{2} \rfloor, j, \lfloor \tfrac{j}{2} \rfloor), \text{ et } c_k = (\lfloor \tfrac{k}{2} \rfloor, \lfloor \tfrac{k}{2} \rfloor, k)$$

Nous définissons le *rang σ induit par le squelette* comme précédemment. Pour chaque site $u = (i,j,k)$, on note : $\sigma_X(u) = ||a_i||_1 + ||u - a_i||_1$, $\sigma_Y(u) = ||b_j||_1 + ||u - b_j||_1$ et $\sigma_Z(u) = ||c_k||_1 + ||u - c_k||_1$, les distances de u aux trois sites du squelette qui ouvriront les trois plans perpendiculaires aux axes, correspondants à chacune des coordonnées de u. Le rang induit par le squelette est défini comme la distance maximum aux trois branches du squelette :

$$\sigma(u) = \max(\sigma_X(u), \sigma_Y(u), \sigma_Z(u))$$

Par construction, $\sigma(u)$ est un minorant du rang dans l'ordre induit par tout jeu de tuiles ordonné ayant pour squelette (a_i), (b_j) et (c_k). $\sigma(u)$ est également un minorant de la date de la pose de la tuile de u pour un tel jeu de tuiles dans la dynamique parallèle.

Nous définissons l'*ordre induit par le squelette* entre deux sites voisins du cube ainsi : pour toute paire de sites voisins u et v, on dit que u est le *prédécesseur* de v (v étant alors *successeur* de u) si $\sigma(u) < \sigma(v)$. On note $\prec$ la fermeture transitive : $u \prec v$ s'il existe une suite finie $(u = u_0, u_1, \dots, u_q = v)$ telle que u_{j-1} est le prédécesseur de u_j pour tout $j = 1, \dots, q$.

thèse [37], cependant il est beaucoup plus simple et efficace ici d'utiliser l'ordre de précédence local dont nous disposons explicitement.

Notre but est de démontrer qu'il existe un jeu de tuiles dont $\prec$ est l'ordre local. Pour cela, nous devons commencer par comprendre la structure de $\prec$ et donc, dans un premier temps, les variations de la fonction σ.

5.2 Variations du rang induit par le squelette

On désigne par $(\vec{x}, \vec{y}, \vec{z})$ la base canonique de $\mathbb{Z}^3$.

Lemme 4.10 (Variations de σ en fonction de z, [39]). *Pour tout $u = (i, j, k) \in \mathbb{N}^3$,*

- *si* $\max(i, j) \leqslant 2k + 1$, *alors* $\sigma(u) < \sigma(u + \vec{z})$;
- *si* $\max(i, j) \geqslant 2k$, *alors* $\sigma(u) < \sigma(u - \vec{z})$. *(à condition que $k \neq 0$)*

La preuve de ce lemme clé repose sur une analyse soigneuse des différents cas possibles. Nous définissons alors les trois surfaces discrètes suivantes, chacune ressemblant à un quart de pyramide aztèque de pente $\frac{1}{2}$:

$$\Delta_X = \{(\lfloor \tfrac{\max(j,k)}{2} \rfloor, j, k) : j, k \geqslant 0\},$$
$$\Delta_Y = \{(i, \lfloor \tfrac{\max(i,k)}{2} \rfloor, k) : i, k \geqslant 0\}, \text{ et}$$
$$\Delta_Z = \{(i, j, \lfloor \tfrac{\max(i,j)}{2} \rfloor) : i, j \geqslant 0\}.$$

On peut alors reformuler le lemme 4.10 ainsi : σ est une fonction strictement croissante de z au dessus de la surface Δ_Z, et strictement décroissante de z en dessous de Δ_Z. Bien sûr, nous avons le même résultat par symétrie pour σ en fonction de x et y par rapport aux surfaces Δ_X et Δ_Y, respectivement.

La figure 4.6 présente ce résultat clé graphiquement.[59] Les surfaces discrètes Δ_X, Δ_Y et Δ_Z sont respectivement de couleurs verte et orange, bleue et rose, bleue et verte (la signification des couleurs sera expliquée à la section suivante). Le sens des variations de σ suivant chaque axe de coordonnées, donné par le lemme 4.10, est représenté par :

- les grosses flèches blanches au sein de chacune des sept régions découpées par les trois surfaces discrètes Δ_X, Δ_Y et Δ_Z ;
- par de petites flèches pour les variations à l'intérieur de chacune de ces trois surfaces discrètes.

Nous en déduisons donc le corollaire suivant du lemme 4.10.

Corollaire 4.11. *σ induit un ordre local strict : pour toute paire de sites voisins u et v dans $\mathbb{N}^3$, nous avons $\sigma(u) \neq \sigma(v)$, c.-à-d. $u \prec v$ ou $v \prec u$. De plus, le rang maximum d'un site de C_{n+1} vaut $3n = ||C_{n+1}||_1$ et est atteint au coin opposé à l'origine, (n, n, n). Ainsi, l'ordre $\prec$ induit par le squelette est bien temps-réel.*

5.3 Classifications des sites en fonction des positions relatives de leurs prédécesseurs

Nous déduisons du lemme 4.10 et de l'observation de la figure 4.6, les propriétés suivantes sur la relation d'ordre $\prec$ induite par σ :

[59] La réalisation de cette figure a été pour moi une étape indispensable dans la compréhension de la relation d'ordre $\prec$.

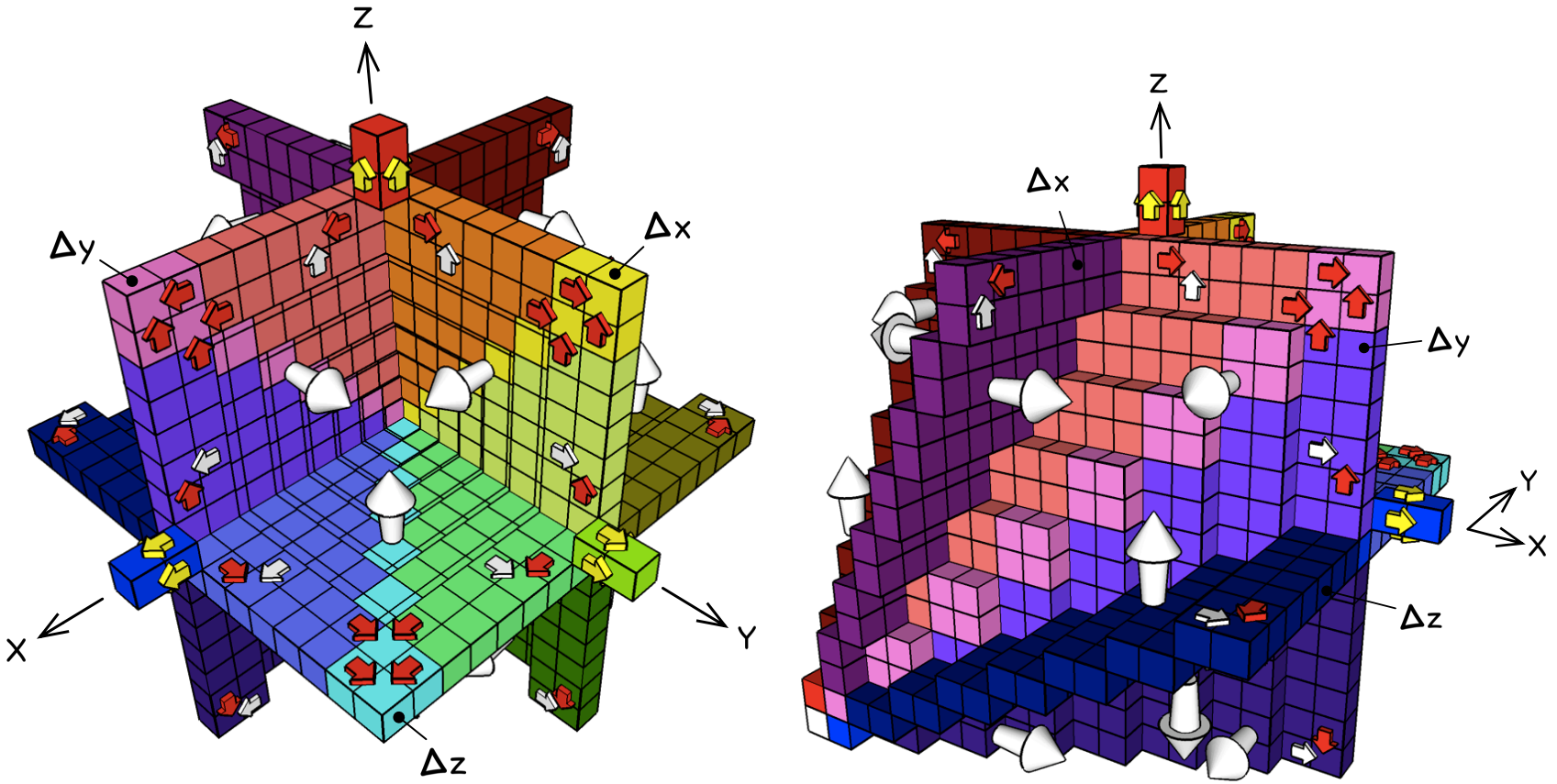


Figure 4.6 – **Vues sous deux angles différents (depuis (n, n, n) à gauche et depuis $(n, 0, n)$ à droite) des variations locales de σ et de l'ordre local induit sur $\mathbb{N}^3$.** Seuls les sites ayant au plus deux prédécesseurs sont représentés (tous les autres en ont trois). Les flèches pointent dans les directions de la croissance de σ selon l'axe de la flèche. Les couleurs des flèches correspondent au type de relation de prédécesseurs : blanche pour un prédécesseur simple, rouge pour un double et jaune pour un triple. L'espace est clairement partitionné en sept régions ne contenant que des sites à trois prédécesseurs simples, séparées par les surfaces discrètes Δ_X, Δ_Y et Δ_Z, elles-mêmes constituées de sites à deux prédécesseurs dont au moins un double, s'intersectant deux par deux suivant les squelettes-x, -y et -z (représentés dans les trois couleurs primaires, bleu, vert et rouge, respectivement) constitués uniquement de sites ayant un unique prédécesseur, triple. *(cette figure a été réalisée avec le logiciel* `SketchUp` *[291])*

Fait 4.12. (a) *Chaque site a au plus trois prédécesseurs parmi ses voisins, tous suivant des axes différents.*

(b) *L'origine est le seul site sans prédécesseur.*

(c) *Un site u a un unique prédécesseur si et seulement si il appartient au squelette. Ce prédécesseur est $u - \vec{z}$ si u appartient au squelette-z (idem par symétrie pour x et y).*

(d) *Tous les sites n'appartenant pas à $\Delta_X \cup \Delta_Y \cup \Delta_Z$ ont exactement trois prédécesseurs.*

(e) *Tout site u ayant deux prédécesseurs a exactement deux voisins sur des faces opposées qui sont ses successeurs. Ses deux successeurs opposés sont $u \pm e_z$ si et seulement si u appartient à Δ_Z (idem par symétrie pour x et y).*

Multiplicité d'un prédécesseur. La notion suivante va nous servir à définir la force des colles liant chaque tuile à ses voisines. La *multiplicité* des prédécesseurs d'un site est définie comme suit (les cas manquants sont obtenus par symétries) :

- la multiplicité des prédécesseurs de tout site à trois prédécesseurs vaut 1 ;
- la multiplicité de l'unique prédécesseur d'un site $u \neq (0,0,0)$ du squelette vaut 3 ;
- Si $u = (i,j,k) \in \Delta_Z$:
  - si $\lfloor \frac{i}{2} \rfloor \geqslant \lfloor \frac{j}{2} \rfloor$ (c.-à-d., si u appartient aux parties bleutées de Δ_Z, fig. 4.6), la multiplicité de son unique prédécesseur suivant l'axe y vaut 2 ;
  - si $\lfloor \frac{j}{2} \rfloor \geqslant \lfloor \frac{i}{2} \rfloor$ (c.-à-d., si u appartient aux parties verdâtres de Δ_Z, fig. 4.6), la multiplicité de son unique prédécesseur suivant l'axe x vaut 2 ;
  - si les deux inégalités précédentes sont vérifiées simultanément (c.-à-d., si u appartient à la partie turquoise sur la diagonale de Δ_Z, fig. 4.6), les multiplicités des deux prédécesseurs de u valent 2 ; sinon la multiplicité de l'autre prédécesseur est 1.

Dans la figure 4.6, les multiplicités des prédécesseurs sont représentées par la couleur des flèches pointant vers leur successeur : blanc, rouge et jaune pour les multiplicités 1, 2, et 3 respectivement. Le fait suivant va garantir que le jeu de tuiles que nous allons déduire de $\prec$ à la section 5.5, assemble correctement à température 3, les productions prescrites par la relation d'ordre.

Fait 4.13. *Pour tout $u \neq (0,0,0)$, la somme des multiplicités de ses prédécesseurs vaut au moins 3, et la somme des multiplicités de sous-ensemble strict de ses prédécesseurs vaut au plus 2.*

5.4 Déduire les successeurs des prédécesseurs

Il s'agit à présent de franchir la dernière étape pour en déduire le jeu de tuile : la capacité à prévoir les successeurs en fonction des prédécesseurs, c.-à-d. la détermination des colles sur les faces ouvertes d'une tuile, en fonction des seules "informations" dont la tuile dispose, les colles de ses faces arrimées à ses tuiles prédécesseurs. En terme de relation d'ordre, il s'agit de démontrer que l'on peut déduire correctement le type des successeurs d'un site u en fonction des types de ses prédécesseurs plus une quantité constante d'information qui peut être transmise de proche en proche par les colles.

Proposition 4.14. *Tout site $u = (i, j, k)$ est dans l'un des cas suivants (à symétrie près) :*

(a) u a un prédécesseur triple, disons $u - \vec{x}$ sans perte de généralité : *alors $u = a_i$ appartient au squelette-x et a quatre successeurs doubles $u \pm \vec{y}$ et $u \pm \vec{z}$; et si i est pair, $u + \vec{x}$ est un successeur triple ($u + \vec{x} = a_{i+1}$), et simple sinon.*

(b) u a deux prédécesseurs doubles, disons $u - \vec{x}$ et $u - \vec{y}$ sans perte de généralité (u appartient à la partie diagonale turquoise de Δ_Z) : *alors $u \pm \vec{z}$ sont des successeurs simples ; et si i (resp. j) est pair, $u + \vec{x}$ (resp. $u + \vec{y}$) est un successeur double, et simple sinon.*

(c) u a deux prédécesseurs, un double et un simple, disons $u - \vec{x}$ et $u - \vec{y}$ respectivement sans perte de généralité (u appartient à la partie vert clair de Δ_Z) : *alors $u \pm \vec{z}$ sont des successeurs simples ; et ses successeurs $u + \vec{x}$ et $u + \vec{y}$ sont respectivement double et simple.*

(d) u a trois prédécesseurs simples : *alors soit u a trois successeurs simples ; soit l'un d'entre eux appartient à une branche du squelette, disons que c'est a_{i+1} sans perte de généralité, alors c'est un successeur triple et les deux autres sont simples, ce qui se produit uniquement si $u = a_i + \vec{y} + \vec{z}$, c.-à-d. si u est le successeur de $a_i + \vec{y}$, lui-même successeur de a_i.*

Chacun de ces cas correspond à une couleur différente sur la figure 4.6. La preuve de cette proposition est une application directe des résultats de la section précédente.

5.5 Jeu de tuiles temps-réel pour cubes

La proposition 4.14 implique que l'on peut correctement déteminer le type des successeurs de tout site à partir de la connaissance de :

1. les multiplicités et positions relatives de ses prédécesseurs,
2. la parité des coordonnées correspondantes, et
3. l'éventuelle proximité d'une des branches du squelette.

Nous définissons alors chaque colle comme un triplet d'étiquettes $(\alpha_1, \alpha_2, \alpha_3)$ définies comme suit :

- Chaque face porte une étiquette $\alpha_1 \in \{+++, ++, +, -, --, ---\}$ suivant sa multiplicité avec le voisin prédécesseur/successeur correspondant et l'orientation de cette relation par rapport à l'axe correspondant.
- Toutes les paires de faces opposées d'une même tuile portent des étiquettes $\alpha_2 \in \{0, 1\}$ opposées mémorisant la parité de la coordonnée correspondante.
- Tout site impair a_{2i+1} du squelette-x indique via α_3 à son successeur $a_{2i+1} + \vec{y}$ qui à son tour informe via α_3 son successeur $a_{2i+1} + \vec{y} + \vec{z}$ que le successeur sur la face x^+ de ce dernier sera le site suivant, a_{2i+2}, du squelette-x et sera donc triple (idem pour les squelettes-y et -z par symétrie).

D'après ce qui précède, nous disposons maintenant d'un jeu de tuiles qui s'auto-assemble depuis la tuile-amorce suivant l'ordre local désiré $\prec$ induit par σ, garant d'une construction en temps-réel. Il ne reste plus qu'à régler le problème de la synchronisation des trois signaux portant les trois branches du squelette. Nous procédons de la même façon que pour les carrés de la section 4.2 en remarquant que tous les sites de la zone centrale du cube, correspondant

à l'intersection des trois demi-espaces définis par les surfaces Δ_X, Δ_Y et Δ_Z, sont tous *à l'heure* (v. définition 4.7). Un calcul rapide démontre que cette région est en fait l'union des cubes $\{q, \ldots, 2q\}^3$ de côtés $q \geqslant 1$, et peut donc être utilisée pour véhiculer un signal de synchronisation `GO`, `STOP-pair` ou `STOP-impair` depuis la diagonale principale jusqu'aux trois branches du squelette en arrivant *juste à temps* pour leur permettre de se décaler d'un cran, de la même façon que pour le carré. On retrouve alors le même phénomène de signaux interdépendants : le signal central de synchronisation a besoin pour avancer des trois autres qui ouvrent les coordonnées et qui, à leur tour, ont besoin de lui pour se décaler. Nous pouvons maintenant conclure.

Théorème 4.15 ([39]). *Il existe un jeu de tuiles auto-assemblantes ordonné et fini qui assemble en temps réel, $3n - 3$, la famille des cubes de côté $n \geqslant 2$ à température 3.*

La figure 4.7 presente quelques images des différents stades de l'auto-assemblage en temps réel du cube avec notre jeu de tuiles. Ces images sont issues de captures d'écran du simulateur que j'ai écrit en `OpenGL`.[60] On peut observer en violet sur l'image de droite de la deuxième ligne, la section hexagonale de la zone des sites *à l'heure* qui propagent les signaux de synchronisation. On remarquera que la construction du cube procède par vagues d'épaisseur 2 et à distance 2 les unes des autres, correspondant au mode de déplacement des branches du squelette : deux pas en avant pour deux sur le côté.

6 Conclusions et perspectives

Nous avons présenté dans ce chapitre un paradigme *top-bottom* pour la conception de jeux de tuiles auto-assemblantes. Ce paradigme fonctionne en deux temps : (1) apposer un ordre local de précédence entre les sites, qui satisfait les propriétés temporelles souhaitées ; et (2) vérifier qu'il vérifie également des propriétés locales qui permettent d'en déduire un jeu de tuiles s'auto-assemblant exactement dans l'ordre prescrit. Cette méthode à l'avantage de découper la conception d'un jeu de tuiles en une suite d'étapes relativement légères. Nous n'aurions sans doute jamais pu obtenir de jeu de tuiles 3D pour construire le cube en temps réel, en procédant de manière frontale à partir du jeu de tuile, ne serait-ce qu'à cause de la complexité de la description des tuiles si l'on courcircuite l'étude de la relation d'ordre sous-jacente.

Le domaine de l'auto-assemblage est assez neuf pour moi mais me semble très intéressant de part ses points communs et ses différences avec d'autres modèles très proches comme les automates cellulaires, les pavages et l'algorithmique distribuée. Parmi les extensions évidentes de nos travaux, figure la recherche de constructions 3D d'autres objets, comme les polyèdres convexes. Florent Becker propose dans sa thèse [37] une construction automatisée des polygones convexes. Sa construction utilise des signaux qui partagent le plan en deux. Or, ce n'est plus possible en 3D et les quelques tentatives que nous avons menées semblent indiquer pour l'instant que la situation se complique singulièrement en dimension 3. En dimension 2, le résultat de [9, 296] démontre que pour auto-assembler une forme, il suffit de $O(K/\log K)$ tuiles, où K est sa complexité de Kolmogorov. Cependant, ce résultat ne dit rien du cas où l'on souhaite fabriquer cette forme à toutes les échelles, comme nous l'avons fait avec le carré. La question de l'auto-assemblage de familles d'objets plus compliqués comme celle des boules euclidiennes discrètes est ouverte. Les constructions connues de cercles discrets par automates cellulaires reposent sur des signaux faisant des allers-et-retours, chose impossible dans le cadre

[60] L'écriture de ce logiciel a été aussi une étape décisive pour notre compréhension de la faisabilité de cette construction.

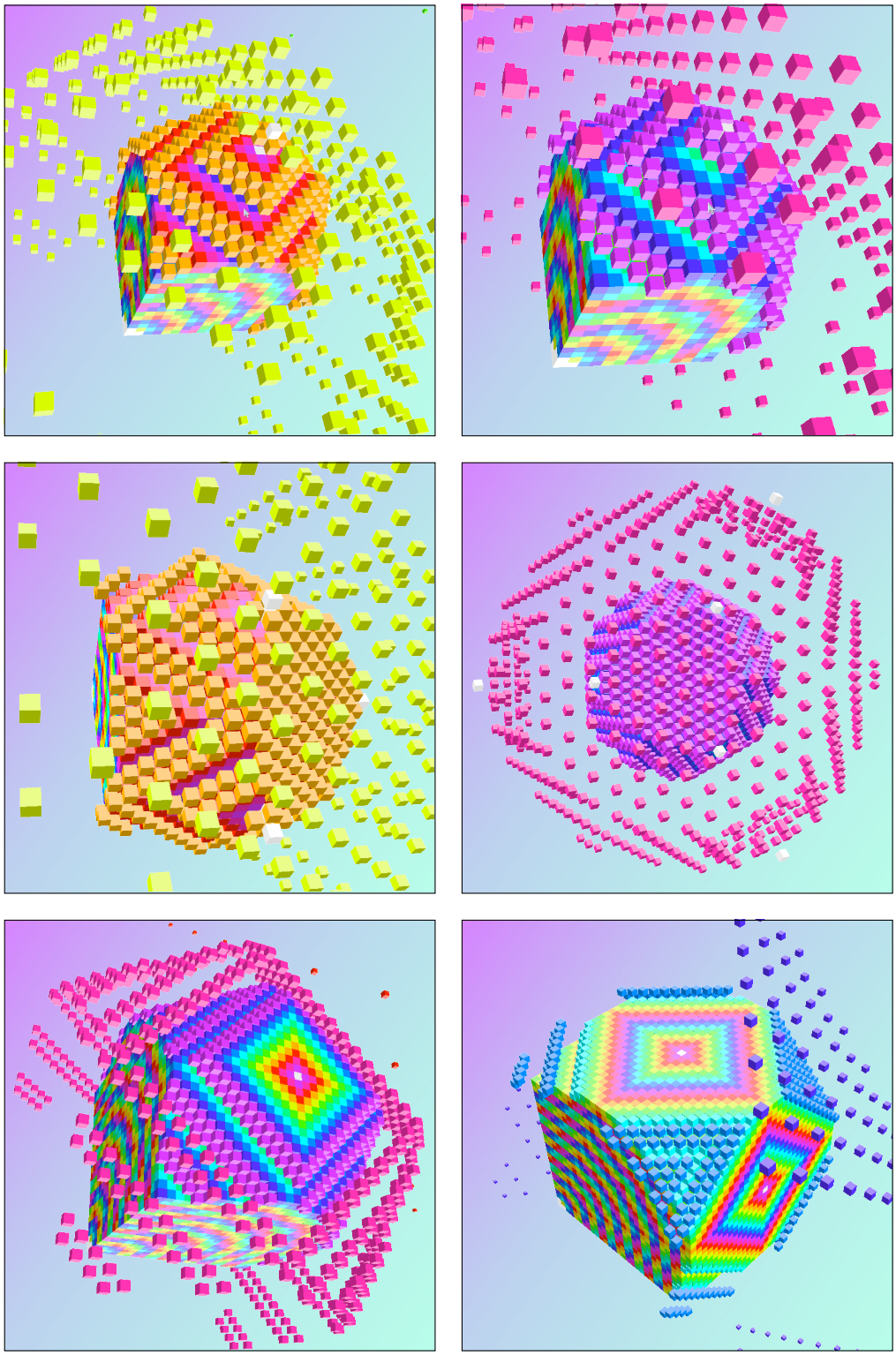
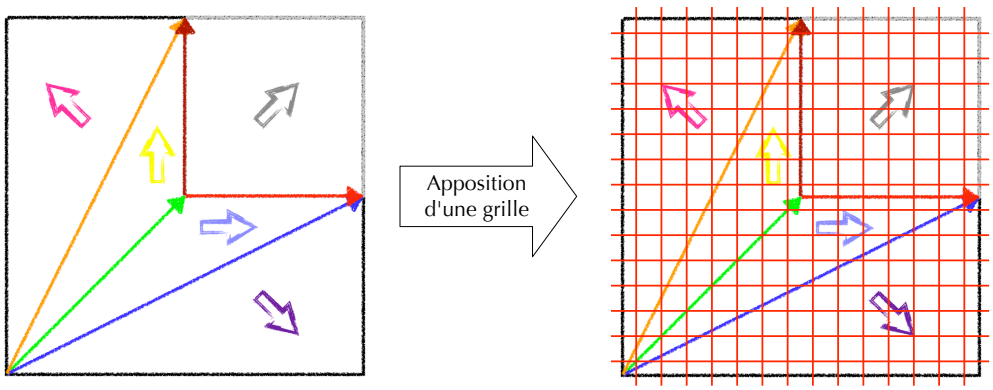


Figure 4.7 – Quelques images des différentes étapes de notre assemblage en temps réel du cube selon la dynamique parallèle. Chaque couleur représente le rang de la tuile modulo 10. Le squelette est répresenté en blanc. De haut en bas et de gauche à droite, les tuiles de rang $\leqslant t$ sont représentées pour des valeurs croissantes de t. (Captures d'écran d'un simulateur de l'assemblage en temps-réel du cube écrit en `OpenGL` [233])

de l'auto-assemblage. L'étude des flux d'information via l'ordre sous-jacent est une piste qu'il conviendrait d'explorer plus en avant.

L'étape de la conception du jeu de tuiles reste fastidieuse et difficile à optimiser. Cette phase d'optimisation est critique car il est très difficile expérimentalement d'obtenir beaucoup de colles, suffisamment différentes pour éviter les erreurs. Florent Becker propose dans sa thèse [37] un langage de programmation à base de signaux où le compilateur engendre un jeu de tuiles à partir de toutes les collisions possibles. Il est nécessaire d'envisager toutes les collisions possibles car déterminer si une collision aura lieu ou non entre deux signaux est indécidable. Or cette étape produit de nombreuses tuiles inutiles. Il me semble qu'il serait intéressant de considérer une approche alternative procédant directement à partir de l'esquisse complète de la construction plutôt qu'à partir des signaux pris indépendamment. La figure 4.8 présente le schéma de cette approche sur le cas de la construction en temps réel des carrés de la section 4.2. L'idée serait de partir du schéma complet de la construction (avec ses signaux et leurs collisions en place et où chaque région est annotée par les directions de propagation) pour le discrétiser à des échelles croissantes jusqu'à ce qu'une "régularité suffisante" soit détectée (p. ex. une periodicité). L'avantage serait alors que seules les tuiles réellement utiles seraient engendrées (p. ex., les tuiles traitant de l'intersection des signaux bleu et orange seraient évitées).

Le lecteur intéressé par ce domaine pourra consulter la thèse de Florent Becker et sa conclusion qui contient de nombreuses questions ouvertes pertinentes.



Esquisse des signaux en place

Découpage de l'esquisse en tuiles par le zoom arrière d'une grille jusqu'à ce qu'une régularité apparaisse

Figure 4.8 – Principe idéal d'une compilation à partir d'une esquisse de la construction.

5 Conclusion

Extrait de [40]
par Benacquista & Barral

Nous arrivons à présent à la conclusion de ce manuscrit. Bien sûr, rien de tout cela n'aurait été possible sans le travail remarquable des étudiants avec lesquels j'ai eu la chance de collaborer : Emmanuelle Lebhar, Damien Regnault, Julien Robert, Sandeep Dey, Nazim Fatès et Florent Becker. Leur inconscience m'a encouragé à me lancer dans des domaines qui m'attiraient mais que je ne connaissais pas toujours bien et dans lesquels je n'avais pas la moindre idée de nos chances de réussite. Cette stratégie a été bénéfique de tout point de vue et je les remercie encore chaudement d'avoir (bien involontairement) permis ces prises de risque, que je jugeais sans doute inconsidérées à l'époque, mais que je n'hésiterai plus à reprendre par la suite.

Nous avons déjà vu un certain nombre de perspectives possibles à nos travaux à la fin de chaque chapitre. Je n'en reprendrai ici que quelques unes.

1 Perspectives dans la continuité de nos travaux

Graphes petits-mondes. Nous avons démontré au chapitre 1 de ce document que l'on pouvait améliorer nos résultats de [195, 196] pour obtenir une meilleure borne, $O(\log n \log\log n)$ (théorème 1.22), sur l'espérance des longueurs des chemins calculés par un algorithme décentralisé dans les graphes de type Kleinberg, c.-à-d. sur leur "diamètre navigable". Or, le diamètre

de ces graphes est connu : $\Theta(\log n)$ [206]. Je suis à peu près convaincu que notre nouvelle borne sur les performances d'un algorithme décentralisé de routage est la meilleure possible. Il serait intéressant de le démontrer, afin de mieux cerner les contre-performances induites par la vision nécessairement limitée des acteurs d'un réseau social.[61]

Je crois que le problème de l'émergence du phénomène petit monde reste une question importante et seulement partiellement résolue : par notre algorithme, efficace mais improbable [90] (v. section 5 du chapitre 1), et par celui de [60] plausible mais trop lent. Une direction à creuser serait celle d'un algorithme dont la construction tiendrait compte de l'évolution des positions des contacts à longue distance pour le déplacement des contacts. Cependant les premières tentatives semblent indiquer que l'analyse s'avère bien plus délicate.

Temps de relaxation et distributions invariantes dans les automates cellulaires. Nous avons observé des transitions de phases nettes polynomial/exponentiel du temps de relaxation pour certains automates élementaires (1D), dont **BCDEFG**(**178**), en fonction du taux d'asynchronisme α sur des configurations finies toriques. Il se trouve que des transitions de phases sur mesures invariantes triviales/non-triviales de cet automate ont été conjecturées simultanément par [205] pour des configurations infinies. Nous sommes à peu près convaincus qu'il s'agit du même phénomène, observé de façons différentes. Il serait très intéressant de réussir à démontrer l'équivalence entre ces deux propriétés.

L'étude de ces petits automates est une étape que je crois indispensable avant de s'attaquer à de plus gros. Je crois aussi à la nécessité de mener ces études de façon aussi exhaustive que possible pour découvrir, peut-être, des comportements insoupçonnés. Cette étude pose la question de l'obtention et de l'écriture des preuves de leur comportement. Nous n'avons fait qu'entrevoir dans ce document l'étonnante richesse des comportements des automates totalisants 2D. Leur caractérisation nécessitera sans aucun doute de nombreuses études de cas, qu'il faudra réussir à écrire et à manipuler pour les conduire. Nous sommes convaincus que cela passera par l'automatisation d'un certain nombre d'étapes en utilisant des formalismes à définir. Les arbres de masques seront sans doute une source d'inspiration intéressante dans cette direction. Ils pourront servir de terrain d'essai à la réalisation d'un assistant pour la conception de fonctions potentiel, dont il pourrait optimiser les choix des poids, p. ex.. Deux logiciels se prêtent sans doute à ces développements, l'assistant de preuve `Coq` et le logiciel de calcul formel `Maple`. Chacun a ses avantages et ses inconvénients et il me semble délicat de trancher pour l'instant.

L'un des obstacles à franchir également, est le choix des fonctionnelles à observer pour étudier convenablement ces comportements. Comme nous l'avons vu au chapitre 2 et tout particulièrement à la section 4.2, ce choix est rendu délicat par la complexité des simulations de ces automates et le fait que l'on ne puisse pas observer directement les configurations mais que l'on doive passer par des projections, c.-à-d. la mesure de fonctionnelles décidées a priori (p. ex., l'énergie ou le nombre de sites actifs). L'obtention de fonctionnelles "universelles" au sens qu'elles permettraient de suivre le comportement de la plupart des automates, sera une grande avancée vers l'obtention d'une carte des comportements possibles et sans doute aussi vers l'automatisation des preuves.

Une direction à explorer pourrait être l'influence de la topologie de l'automate sur ses motifs dominants. Nous avons en effet observé que les motifs émergents dans les automates

[61] Avec George Giakkoupis, nous avons par la suite apporté une réponse complète à cette question, publiée dans [134] : on ne peut effectivement pas faire mieux que $\Omega(\log n \log\log n)$ en dimension 1 en ne visitant qu'un nombre polylogarithmique de nœuds ; mais on peut trouver des chemins de longueur optimale $O(\log n)$ dès que la dimension de la grille sous-jacente est $\geqslant 2$.

totalisants ne sont pas si nombreux et semblent être liés aux relations de voisinage : p. ex., les damiers sont très courants pour le voisinage de von Neumann, mais nous ne les avons pas rencontrés dans nos simulations d'automates totalisants avec le voisinage de Moore. La confirmation de ces faits pourraient aider à construire des fonctionnelles génériques.

Ordonnancement non clairvoyant. Le chapitre 3 est dédié à l'étude d'une fonction objectif : le *temps de service moyen*. D'un côté, cette fonction est la plus naturelle au sens qu'elle correspond au temps moyen d'attente d'un client ; mais de l'autre, cette fonction encourage des comportements que l'on peut qualifier de déviants de la part des algorithmes. En effet, comme nous l'avons vu, la seule chose qui compte pour le temps de flot, est le nombre de requêtes actives à chaque instant, quelques soient ces requêtes. Ceci pousse au développement d'algorithmes qui tendent à privilégier systématiquement les requêtes les récentes, quitte à ne jamais servir les requêtes les plus anciennes, comme **LAPS**$_\beta$. Il serait intéressant d'étudier si cet écueil est incontournable, ou si d'autres stratégies (randomisées, p. ex.) pourraient l'éviter.

La recherche d'un *unique* algorithme non-clairvoyant compétitif à toute vitesse $s > 1$ est une question ouverte particulièrement intéressante (**LAPS**$_\beta$ n'est compétitif que pour une vitesse $s > 1 + \beta$). Jeff Edmonds propose dans [92] un schéma de preuve tendant à démontrer que ce n'est pas possible. Les techniques proposées sont très originales (elles utilisent le théorème de Brouwer) et méritent toute notre attention.

L'étude du *broadcast* est un sujet très actif en ordonnancement. L'introduction de dépendances plus sophistiquées que des requêtes pour des ensembles d'éléments, semble compliquer considérablement la donne. Il serait intéressant de voir si l'introduction de ces dépendances modifie ou non la charge maximale supportée par le serveur (nous craignons que oui). Une possibilité serait de tenter de définir une alternative au coefficient d'étalement, adaptée au cas du *broadcast*.

Auto-assemblage. Nous avons vu qu'il peut être instructif de découpler le processus d'assemblage d'un objet de celui du calcul de sa taille. Nous voyons alors surgir des contraintes originales entre le processus d'auto-assemblage local et la géométrie globale de l'objet. Ces contraintes imposées sont encore très mal comprises. Nous avons vu que l'étude des relations d'ordre local pouvait aider à y voir plus clair, mais nous pensons qu'il faudrait les explorer plus avant. Le cas de la construction de la famille des disques discrets par auto-assemblage nous semble être un point de départ intéressant. À notre connaissance, il n'existe pas de construction de ces disques dont les signaux ne fassent pas d'allers-et-retours à l'intérieur de la forme, chose interdite en auto-assemblage. Ces formes sont donc de bonnes candidates soit pour ne pas être auto-assemblables, soit pour rechercher de nouvelles constructions originales.

Florent Becker propose dans sa thèse [37] un procédé de génération automatique d'un jeu de tuiles à partir de systèmes de signaux, sans rien présumer du schéma global de la construction sous-jacente. Nous pensons cependant que d'autres procédés s'aidant de ce schéma pourraient être développés (v. section 6 du chapitre 4). Ils présenteraient sans doute l'avantage de réduire considérablement le nombre de tuiles générées.

2 Pour aller plus loin

Nous avons donc vu dans ce manuscrit qu'il était possible d'utiliser des méthodes algorithmiques pour étudier des phénomènes relevant naturellement des systèmes dits complexes. Je vous propose dans cette section un certain nombre de directions qui me semblent permettre d'aller plus loin dans ce paradigme.

Vers une définition de la complexité. Nous avons vu dans le chapitre introductif de ce document que la mise en œuvre de la complexité de Kolmogorov-Solomonoff-Chaitin (section 1.7 du chapitre 0) posait de nombreuses difficultés. Outre le fait qu'elle ne soit pas calculable, elle ne permet pas de qualifier de complexes la plupart des systèmes que l'on aimerait y rattacher, ni de donner un sens au "hasard déterministe" mis en évidence par Poincaré. J'évoquais alors à la section 3.2.4 (chapitre 0) les résultats récents en cryptographie sur la dérandomisation : il est possible de construire des suites de bits pseudo-aléatoires à partir de fonctions difficiles à calculer, sur lesquelles tout algorithme polynomial se comporte asymptotiquement de la même façon que si c'était une suite de bits purement aléatoires ; il y a de plus équivalence : de telles suites existent si et seulement si il existe une fonction vraiment difficile à calculer (une fonction de EXP qui ne peut pas être calculée par des circuits non-uniformes de taille polynomiale). Ainsi, l'existence même du hasard déterministe serait liée à des conjectures traditionnelles en théorie de la complexité algorithmique. Il me semble qu'il doit être possible d'utiliser ce résultat pour préciser le sens de l'expression "systèmes complexes". Je vois deux directions à explorer. D'une part, je propose d'étudier certains systèmes pour démontrer qu'ils sont suffisamment simples (de taille polynomiale) pour que des suites pseudo-aléatoires suffisent à en expliquer la complexité ; et donc valider ainsi la thèse que le chaos suffit à expliquer le bon fonctionnement de la théorie des probabilités pour l'analyse de certains phénomènes en pratique. D'autre part, tenter d'utiliser ces travaux sur l'extraction et la dilution de l'aléatoire en cryptographie, pour tenter de définir une mesure de la quantité d'aléatoire présente dans certains phénomènes dits complexes. Comment vous pouvez le voir, ces idées sont encore assez floues, mais je reste convaincu que les résultats obtenus en cryptographie sont traduisibles en terme de systèmes complexes. Le semestre que nous organisons à l'IHP en 2011 avec Pierre Pansu autour des progrès récents en théorie de la complexité algorithmique et leur convergence avec certaines branches de la mathématique, sera certainement l'occasion d'étudier plus profondément ces questions.

Informer et s'informer. Comme je l'ai fait remarquer (et peut-être démontré) dans l'introduction de ce manuscrit, je suis convaincu que de nombreux concepts développés dans chaque domaine scientifique ont une portée bien plus large que celle de leur propre domaine. Aussi, il me semble très important de proposer des formulations claires des résultats que nous jugeons importants, pour qu'ils puissent être accessibles à d'autres ; et, dans le même temps, d'être attentif aux progrès qui se font dans les différents domaines qui nous entourent. Pour le premier point, j'ai toujours accordé une part importante de mon activité à la vulgarisation. Elle permet à la fois de clarifier les idées que l'on souhaite expliquer, et ensuite de les confronter aux points de vue de personnes ayant une autre culture et qui peuvent en retour nous aider à enrichir nos concepts. Pour le second point, je dois avouer que l'existence d'organismes comme le CNRS où la plupart des sciences cohabitent et où nous sommes régulièrement informés de qui se passe dans chacun de ces domaines est une grande chance sans laquelle je n'aurais pas pu développer mes recherches de la même façon.

Bibliographie

[1] Ittai Abraham, Yair Bartal, Hubert T.-H. Chan, Kedar Dhamdhere, Anupam Gupta, Jon M. Kleinberg, Ofer Neiman, and Aleksandrs Slivkins. Metric embeddings with relaxed guarantees. In *Proc. of IEEE Conf. on Foundations of Computer Science (FOCS)*, pages 83–100, 2005.

[2] Ittai Abraham, Yair Bartal, and Ofer Neiman. Embedding metric spaces in their intrinsic dimension. In *Proc. of ACM/SIAM Symp. On Discrete Algorithms (SODA)*, pages 363–372, 2008.

[3] Ittai Abraham and Cyril Gavoille. Object location using path separators. In *Proc. of ACM Symp. on Principles of Distributed Computing (PODC)*, pages 188–197, 2006.

[4] Ittai Abraham, Cyril Gavoille, Dahlia Malkhi, Noam Nisan, and Mikkel Thorup. Compact name-independent routing with minimum stretch. In *Proc. of ACM Symp. on Parallel Algorithms and Architectures (SPAA)*, pages 20–24, 2004.

[5] Ittai Abraham and Dahlia Malkhi. Name independent routing for growth bounded networks. In *Proc. of ACM Symp. on Parallel Algorithms and Architectures (SPAA)*, pages 49–55, 2005.

[6] Swarup Acharya. *Broadcast Disks: Dissemination-based Management for Assymmetric Communication Environments*. PhD thesis, Brown University, May 1998.

[7] Swarup Acharya, Michael J. Franklin, and Stanley B. Zdonik. Balancing push and pull for data broadcast. In *Proc. of ACM International Conference on Management of Data (SIGMOD)*, Tuscon, Arizona, Mai 1997.

[8] Dimitris Achlioptas, Aaron Clauset, David Kempe, and Cristopher Moore. On the bias of traceroute sampling; or, power-law degree distributions in regular graphs. In *Proc. of the Symp. on Theory of Computing (STOC)*, 2005. To appear in J. of the ACM.

[9] Leonard M. Adleman, Qi Cheng, Ashish Goel, and Ming-Deh A. Huang. Running time and program size for self-assembled squares. In *Proc. of ACM Symp. on Theory of Computing (STOC)*, pages 740–748, 2001.

[10] J. Adler, D. Stauffer, and A. Aharony. Comparison of bootstrap percolation models. *Journal of Physics A*, 22:L297–L301, 1991.

[11] Réka Albert and Albert-László Barabási. Diameter of the World Wide Web. *Nature*, 401, Sept. 1999.

[12] David Aldous and Jim Fill. *Reversible Markov Chains and Random Walks on Graphs*. Online draft, 1999-2001. http://www.stat.berkeley.edu/~aldous/RWG/book.html.

[13] Aurélien Alvarez, Étienne Ghys, and Jos Leys. *Dimensions*. École normale supérieure de Lyon, 2008. (DVD, 2h). http://www.dimensions-math.org/.

[14] M. H. Ammar. Response time in a teletext system: An individual user's perspective. *IEEE Trans. on Communications*, 35(11):1159–1170, Nov. 1987.

[15] M. H. Ammar and J. W. Wong. The design of teletext broadcast cycles. *Performance Evaluation*, 5(4):235–242, 1985.

[16] Reid Andersen, Fan Chung, and Linyuan Lu. Modeling the small-world phenomenon with local network flow. *Internet Math.*, 2(3):359–385, 2006.

[17] Dan Ariely. Why we think it's OK to cheat and steal (sometimes). In *TED Ideas worth spreading*, March 2009. http://www.ted.com/index.php/talks/dan_ariely_on_our_buggy_moral_code.html.

[18] Sanjeev Arora and Boaz Barak. *Complety theory: a modern approach*. Cambridge University Press, 2009 (à paraître). http://www.cs.princeton.edu/theory/complexity/.

[19] Juan A. Asenjo, P. Ramirez, Iván Rapaport, Julio Aracena, and Eric Goles abd Barbara A. Andrews. A discrete mathematical model applied to genetic regulation and metabolic networks. *J. of Microbiology and Biotechnology*, 17(3):496–510, 2007.

[20] Yonatan Aumann and Yuval Rabani. An $O(\log k)$ approximate min-cut max-flow theorem and approximation algorithm. *SIAM J. of Computing*, 27(1):291–301, 1998.

[21] O. T. Avery, C. M. MacLeod, and M. McCarty. Studies on the chemical nature of the substance inducing transformation of pneumococcal types. *Journal of Experimental Medicine*, 79(2):137–158, 1944.

[22] Yossi Azar, Amos Fiat, Anna R. Karlin, Frank McSherry, and Jared Saia. Spectral analysis of data. In *Proc. of ACM Symp. on Theory of Computing (STOC)*, pages 619–626, 2001.

[23] David Bailey, Peter Borwein, and Simon Plouffe. On the rapid computation of various polylogarithmic constants. *Mathematics of Computation*, 66(218):903–913, 1997.

[24] P. Bak, C. Tang, and K. Wiesenfeld. Self-organized criticality: an explanation of $1/f$ noise. *Physical Review Letters*, 59:381–384, 1987.

[25] Paul N. Balister, Béla Bollobás, and Robert Kozma. Large deviations for mean fields models of probabilistic cellular automata. *Random Structures and Algorithms*, 29(3):399–415, 2006.

[26] József Balogh, Béla Bollobás, and Robert Morris. Majority bootstrap percolation on the hypercube. *TO BE FOUND !!!*, 2007.

[27] N. Bansal, M. Charikar, S. Khanna, and J. S. Naor. Approximation the average response time in broadcast scheduling. In *Proc. of ACM-SIAM SODA*, 2005.

[28] Nikhil Bansal, Don Coppersmith, and Maxim Sviridenko. Improved approximation algorithms for broadcast scheduling. *SIAM J. of Computing*, 38(3):1157–1174, 2008. An abstract version was published in SODA2006.

[29] A. Bar-Noy, R. Bhatia, J. Naor, and B. Schieber. Minimizing service and operation costs of periodic scheduling. In *Proc. of the 9th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA'98)*, pages 11–20, 1998.

[30] Amotz Bar-Noy, Joseph Naor, and Baruch Schieber. Pushing dependent data in clients-providers-servers systems. *Wireless Networks*, 9:421–230, 2003.

[31] Amotz Bar-Noy and Yaron Shilo. Optimal broadcasting of two files over an asymmetric channel. *J. Parallel Distrib. Comput.*, 60(4):474–493, 2000. (also published in the Proc. of INFOCOM 1999).

[32] Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.

[33] L. Barrière, P. Fraigniaud, E. Kranakis, and D. Krizanc. Efficient routing in networks with long range contacts. In *Proc.of the Int. Conf. on Distributed Computing (DISC)*, pages 270–284, 2001.

[34] Bonnie Bassler. Discovering bacteria's amazing communication system. http://www.ted.com/index.php/talks/bonnie_bassler_on_how_bacteria_communicate.html.

[35] Chumki Basu, Haym Hirsh, and William W. Cohen. Recommendation as classification: using social and content-based information in recommendation. In *Proc. of National Conf. on Artificial Intelligence and Innovative Applications of Artificial Intelligence (AAAI/IAAI)*, pages 714–720, 1998.

[36] Luca Becchetti and Stefano Leonardi. Nonclairvoyant scheduling to minimize the total flow time on single and parallel machines. *J. ACM*, 51(4):517–539, 2004. An abstract version was published in STOCS2001.

[37] Florent Becker. *Géométrie pour l'auto-assemblage*. PhD thesis, École normale supérieure de Lyon, 2008. Réf. 08ENSL0495.

[38] Florent Becker, Iván Rapaport, and Éric Rémila. Self-assemblying classes of shapes with a minimum number of tiles, and in optimal time. In *Proc. of Found. of Software Tech. and Theo. Comp. Sci. (FSTTCS)*, volume 4337 of *LNCS*, pages 45–56, 2006.

[39] Florent Becker, Éric Rémila, and Nicolas Schabanel. Time optimal self-assembling of 2D and 3D shapes: the case of squares and cubes. In *Proc. of 14th Int. Meeting on DNA Computing (DNA14)*, volume 5347 of *LNCS*, pages 144–155, June 2008.

[40] Tonino Benacquista and Nicolas Barral. *Dieu n'a pas réponse à tout (mais Il sait à qui s'adresser)*. Dargaud, 2008.

[41] N. Berger, Béla Bollobás, C. Borgs, J. T. Chayes, and O. Riordan. Degree distribution of the FKP network model. In *Proc. of Int. Colloquium on Automata, Languages and Programming (ICALP)*, 2003.

[42] Noam Berger, Christian Borgs, Jennifer T. Chayes, and Amin Saberi. On the spread of viruses on the internet, january 23-25, 2005, vancouver, british columbia. In *Proc. of ACM/SIAM Symp. On Discrete Algorithms (SODA)*, pages 301–310, 2005.

[43] E. R. Berlekamp, John H. Conway, and R. K. Guy. *Winning ways for your mathematical plays*, volume 2, chapter 25 "What is Life?". Academic Press, 1982.

[44] Piotr Berman and Janos Simon. Investigations of fault-tolerant networks of computers (preliminary version). In *Proc. of ACM Symp. on Theory of Computing (STOC)*, pages 66–77, 1988.

[45] Hugues Bersini and Vincent Detours. Asynchrony induces stability in cellular automata based models. In *Proc. of Conf. on Artificial Life*, pages 382–387, 1994.

[46] Azer Bestavros. Speculative data dissemination and service to reduce server load, network traffic and service time in distributed information systems. In *Proc. of the 1996 International Conf. on Data Engineering (ICDE'96)*, Mars 1996.

[47] Manuel Blum, Michael Luby, and Ronnit Rubinfeld. Self-testing/correcting with applications to numerical problems. In *Proc. of ACM Symp. on Theory of Computing (STOC)*, pages 73–83, 1990.

[48] Manuel Blum and Silvio Micali. How to generate cryptographically strong sequences of pseudo-random bits. *SIAM J. of Computing*, 13(4):850–864, 1984. Version préliminaire publiée à FOCS 1982.

[49] Hans L. Bodlaender. Treewidth: Structure and algorithms. In *Proc. of Int. Colloquium on Structural Information and Communication Complexity (SIROCCO)*, pages 11–25, 2007. Invited talk.

[50] B. Bollobás and O. Riordan. In *Handbook of graphs and networks*, chapter Mathematical results on scale-free random graphs. Berlin, 2002.

[51] A. Borodin and R. El-Yaniv. *Online computation and competitive analysis*. Cambridge university press, 1998.

[52] Romain Boulet, Bertrand Jouve, Fabrice Rossi, and Nathalie Villa. Batch kernel SOM and related Laplacian methods for social network analysis. *Neurocomputing*, 71(7-9):1257–1273, 2008. Paru également sous le titre « Social networking gets medieval: researchers give a French province the 'Facebook' treatment ».

[53] Jean Bourgain. On Lipschitz embedding of finite metric spaces in Hilbert spaces. *Israeli J. Math*, 52:46–52, 1985.

[54] Olivier Bournez, Jérémie Chalopin, Johanne Cohen, and Xavier Kœgler. Playing with population protocols. In *Proc. of Complexity of Simple Programs (CSP)*, pages 1–14, 2008.

[55] Anton Bovier and Francesco Manzo. Metastability in Glauber dynamics in the low temperature limit: beyond exponential asymptotics. *Journal of Statistical Physics*, 107:757–779, 2002.

[56] Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks*, 30(1–7):107–117, 1998.

[57] A. Z. Broder. How hard is it to marry at random? (On the approximation of the permanent). In *Proc. of ACM Symp. on Theory of Computing (STOC)*, pages 50–58, 1986. Erratum in *Proc. of ACM Symp. on Theory of Computing (STOC)* (1988), p. 551.

[58] R.L. Buvel and T.E. Ingerson. Structure in asynchronous cellular automata. *Physica D*, 1:59–68, 1984.

[59] J. M. Carlson and J. Doyle. Highly optimized tolerance: a mechanism for power laws in designed systems. *Physics Review E*, 60(2):1412–1427, 1999.

[60] Augustin Chaintreau, Pierre Fraigniaud, and Emmanuelle Lebhar. Networks become navigable as nodes move and forget. In *Proc. of Int. Colloquium on Automata, Languages and Programming (ICALP)*, volume LNCS 5125, pages 133–144, 2008.

[61] G. J. Chaitin. On the length of programs for computing finite binary sequences. *Journal of the ACM*, 13:547–569, 1966.

[62] Jessica Chang, Thomas Erlebach, Renars Gailis, and Samir Khuller. Broadcast scheduling: algorithms and complexity. In *Proc. of ACM/SIAM Symp. On Discrete Algorithms (SODA)*, pages 473–482, 2008.

[63] Philippe Chassaing and Lucas Gérin. Asynchronous cellular automata and brownian motion. In *Proc. of Int. Conf. on Analysis of Algorithms (AofA)*, pages 385–402, 2007.

[64] Brigitte Chauvin. Martingales discrètes et applications à l'analyse d'algorithmes. In *École ALÉA*, CIRM, march 2002.
http://algo.inria.fr/seminars/sem01-02/chauvin.pdf.

[65] Chandra Chekuri, Sungjin Im, and Benjamin Moseley. Longest wait first for broadcast scheduling. In *Proc. of Int. Work. on Approximation and Online Algorithms (WAOA)*, 2009. To appear.

[66] Ho-Lin Chen, Rebecca Schulman, Ashish Goel, and Erik Winfree. Reducing facet nucleation during algorithmic self-assembly. *Nano Letters*, 7(9):2913–2919, 2007.

[67] Fan Chung and Linyuan Lu. The small world phenomenon in hybrid power law graphs. In *Lect. Notes Phys.*, volume 650, pages 89–104, 2004.

[68] Aaron Clauset and Cristopher Moore. How do networks become navigable?
http://arxiv.org/abs/0803.0248.

[69] Aaron Clauset and Cristopher Moore. Accuracy and scaling phenomena in Internet mapping. *Physical Review Letters*, 94(0108701), 2005.

[70] Bram Cohen. BitTorrent, 2001. http://www.bittorrent.com.

[71] Complete viral genomes website.
http://www.genome.jp/dbget-bin/get_htext?Viruses+-n.

[72] M. Cook. Universality in elementary cellular automata. *Complex systems*, 14(1-40), 2004.

[73] D. Coppersmith, D. Gamarnik, and M. Sviridenko. The diameter of a long-range percolation graph. *Random Structures and Algorithms*, 21(1):1–13, 2002.

[74] Bruno Courcelle and Mohamed Mosbah. Monadic second-order evaluations on tree-decomposable graphs. *Theoretical Computer Science*, 109(1-2):49–82, 1993.

[75] James P. Crutchfield and Melanie Mitchell. The evolution of emergent computation. In *Proc. of the Nat. Academy of Science of the USA (PNAS)*, volume 92, pages 10742–10746, 1995.

[76] B. A. Davey and H. A. Priestley. *Introduction to Lattices and Order*. Cambridge University Press, 2002.

[77] Christophe Dejours. *L'évaluation du travail à l'épreuve du réel - Critique des fondements de l'évaluation*. INRA, 2003.

[78] Denis Delbecq. Moyen-Âge, le temps des réseaux. *Le journal du CNRS*, 224:9, 2008.

[79] J. Demongeot, J. Aracena, F. Thuderoz, T.-P. Baum, and O. Cohen. Genetic regulation networks: circuits, regulons and attractors. *C.R. Biologies*, 326:171–188, 2003.

[80] Jacques Demongeot, Christelle Jézéquel, and Sylvain Sené. Boundary conditions and phase transitions in neural networks. theoretical results. *Neural Networks*, 21(7):971–979, 2008.

[81] Jacques Demongeot, Michel Morvan, and Sylvain Sené. Impact of fixed boundary conditions on the basins of attraction in the flowers morphogenesis of arabidopsis thaliana. In *Proc. of Int. Conf. on Advanced Information Networking and Applications (AINA)*, pages 782–789, 2008.

[82] Xiaotie Deng, Christos H. Papadimitriou, and Shmuel Safra. On the complexity of equilibria. In *Proc of ACM Symp. on Theory of Computing (STOC)*, pages 67–71, 2002.

[83] Sandeep Dey and Nicolas Schabanel. Customized newspaper broadcast: data broadcast with dependencies. In *Proc. of the Latin Amerian Symp. on Theoretical Informatics (LATIN)*, volume LNCS 3887, pages 362–373, Mar. 2006.

[84] Robert P. Dilworth. A decomposition theorem for partially ordered sets. *Annals of Mathematics*, 51:161–166, 1950.

[85] Nature archives double helix of DNA: 50 years, 2003. http://www.nature.com/nature/dna50/.

[86] P. S. Dodds, R. Muhamad, and Ducan J. Watts. An experimental study of searching in global social networks. *Science*, 301:827–829, 2003.

[87] Adrien Douady. *La dynamique du lapin*. CERIMES: Centre de ressources et d'information sur les multimédias pour l'enseignement supérieur, 1996. (DVD, 24 min.).

[88] Philippe Duchon, Nicolas Hanusse, Emmanuelle Lebhar, and Nicolas Schabanel. Brief announcement: Could any graph be turned into a small world? In *Proc. of the Int. Symp. on Distributed Computing (DISC)*, volume LNCS 3724, pages 511–513, 2005.

[89] Philippe Duchon, Nicolas Hanusse, Emmanuelle Lebhar, and Nicolas Schabanel. Could any graph be turned into a small world? *Theoretical Computer Science, Special edition on complex networks*, 355(1):96–103, 2006.

[90] Philippe Duchon, Nicolas Hanusse, Emmanuelle Lebhar, and Nicolas Schabanel. Towards small world emergence. In *Proc. of ACM Symp. on Parallel Algorithms and Architectures (SPAA)*, pages 225–232, july 2006.

[91] Bruno Durand and Zs. Róka. The game of life: universality revisited. In *Cellular Automata (Saissac, 1996)*, pages 51–74. Kluwer Academic Publisher, 1999.

[92] Jeff Edmonds. Every deterministic nonclairvoyant scheduler has a suboptimal load threshold. Work in progress (available on his website).

[93] Jeff Edmonds. Scheduling in the dark. *Manuel Blum's Special Issue of Theoretical Computer Science*, 235(1):109–141, 2000. Version préliminaire publiée à STOC 1999.

[94] Jeff Edmonds. *Encyclopedia of Algorithms*, chapter Scheduling with Equipartition, pages 1–99. Springer, 2008.

[95] Jeff Edmonds, Donald D. Chinn, Tim Brecht, and Xiaotie Deng. Non-clairvoyant multiprocessor scheduling of jobs with changing execution characteristics. *J. of Scheduling*, 6(3):231–250, 2003. An abstract version was published in STOCS1997.

[96] Jeff Edmonds, S. Datta, and P. Dymond. TCP is competitive against limited adversary. In *Proc. of ACM Symp. on Parallel Algorithms and Architectures (SPAA)*, pages 174–183, 2003.

[97] Jeff Edmonds and Kirk Pruhs. Multicast pull scheduling: When fairness is fine. *Algorithmica*, 36(3):315–330, 2003. Une version préliminaire a été publiée à SODA2002.

[98] Jeff Edmonds and Kirk Pruhs. A maiden analysis of longest wait first. *ACM Trans. Algorithms*, 1(1):14–32, 2005. An abstract version was published in SODA2004.

[99] Jeff Edmonds and Kirk Pruhs. Scalably scheduling processes with arbitrary speedup curves. In *Proc. of ACM/SIAM Symp. On Discrete Algorithms (SODA)*, pages 685–692, 2009. www.cse.yorku.ca/~jeff/research/schedule/laps.pdf.

[100] F. Eggenberger and G. Pólya. Über die Statistik Verketteter. *Zeitschrift Agnew. Math. Mech.*, 3:279–289, 1923.

[101] D. M. Eigler and E. K. Schweizer. Positioning single atoms with a scanning tunneling microscope. *Nature*, 344(524-526), 1990.

[102] Paul Erdős and A. Rényi. On random graphs. *Publ. Math. (Debrecen)*, pages 290–297, 1959.

[103] Thomas Erlebach and Alexander Hall. NP-hardness of broadcast scheduling and inapproximability of single-source unsplittable min-cost flow. *J. of Scheduling*, 7(3):223–241, 2004. An abstract version was published in SODA2002.

[104] Alex Fabrikant, Elias Koutsoupias, and Christos H. Papadimitriou. Heuristically optimized trade-offs: A new paradigm for power laws in the internet. In *Proc. of Int. Colloquium on Automata, Languages and Programming (ICALP)*, volume 2380, pages 110–122, 2002.

[105] Alex Fabrikant, Christos H. Papadimitriou, and Kunal Talwar. The complexity of pure nash equilibria. In *Proc. of ACM Symp. on Theory of Computing (STOC)*, pages 604–612, 2004.

[106] Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. On power-law relationships of the internet topology. *Comput. Commun. Rev.*, 29:251–262, 1999.

[107] Nazim Fatès. *Robustesse de la dynamique des systèmes discrets : le cas de l'asychronisme dans les automates cellulaires*. PhD thesis, École normale supérieure de Lyon, 2004.

[108] Nazim Fatès and Lucas Gérin. Examples of fast and slow convergence of 2D asynchronous cellular systems. *Journal of Cellular Automata*, 4:323–337, 2009.

[109] Nazim Fatès and Michel Morvan. An experimental study of robustness to asynchronism for elementary cellular automata. *Complex Systems*, 16(1):1–27, 2005.

[110] Nazim Fatès, Michel Morvan, Nicolas Schabanel, and Éric Thierry. Fully asynchronous behavior of double-quiescent elementary cellular automata. In *Proc. of the Symp. on Mathematical Foundation of Computer Science (MFCS)*, volume LNCS 3618, pages 316–327, 2005.

[111] Nazim Fatès, Michel Morvan, Nicolas Schabanel, and Éric Thierry. Fully asynchronous behavior of double-quiescent elementary cellular automata. *Theoretical Computer Science*, 362(1-3):1–16, 2006.

[112] Nazim Fatès, Damien Regnault, Nicolas Schabanel, and Éric Thierry. Asynchronous behavior of double-quiescent elementary cellular automata. In *Proc. of the Latin Amerian Symp. on Theoretical Informatics (LATIN 2006)*, volume 3887, pages 455–466, Mar. 2006.

[113] W. Feller. *An introfuction to probability theory*. John Willey & Sons, 3rd edition, 1968.

[114] Pablo A. Ferrari. Exclusion processes and applications. In *Interacting particles systems, statistical mecanics and probability theory, Semestre à l'Institut Henri Poincaré*, sep.-déc. 2008. Cours disponibles à: http://www.ime.usp.br/∼pablo/papers/ihp2008/ihp2008.pdf.

[115] Philippe Flajolet. Counting by coin tossing (keynote paper). In *LNCS Proc. of Asian Computing Science Conference (ASIAN)*, volume 3321, pages 1–12, 2004.

[116] Philippe Flajolet and Robert Sedgewick. *Analytic combinatorics*. Cambridge University Press, 2008. ISBN: 9780521898065.

[117] Pierre Fraigniaud. Greedy routing in tree-decomposed graphs: a new perspective on the small-world phenomenon. In *Proc. of European Symp. on Algorithms (ESA)*, volume LNCS 3669, pages 791–802, 2005.

[118] Pierre Fraigniaud and Philippe Gauron. Brief announcement: An overview of content-addressable network D2B. In *Proc. of ACM Symp. on Principles of Distributed Computing (PODC)*, page 151, 2003.

[119] Pierre Fraigniaud, Cyril Gavoille, and Christophe Paul. Eclecticism shrinks even small worlds. In *Proc. of ACM Symp. on Principles of Distributed Computing (PODC)*, pages 169–178, 2004.

[120] Pierre Fraigniaud, Emmanuelle Lebhar, and Zvi Lotker. A doubling dimension threshold $\Theta(\log \log n)$ for augmented graph navigability. In *Proc. of European Symp. on Algorithms (ESA)*, volume LNCS 4168, pages 376–386, 2006.

[121] Pierre Fraigniaud, Emmanuelle Lebhar, and Zvi Lotker. Recovering the long range links in augmented graphs. In *Proc. of Int. Colloquium on Structural Information and Communication Complexity (SIROCCO)*, volume LNCS 5058, pages 104–118, 2008.

[122] Pierre Fraigniaud, Emmanuelle Lebhar, and Laurent Viennot. The inframetric model for the Internet. In *Proc. of IEEE Int. Conf. on Computer Communications (INFOCOM)*, pages 1085–1093, April 2008.

[123] Robert H. Frank, Thomas Gilovich, and Dennis T. Regan. Does studying economics inhibit cooperation? *Journal of Economic Perspectives*, 7:159–171, 1993. http://www.gnu.org/philosophy/economics_frank/frank.html.

[124] Marie Frebus-Zanda and Serge Grigorieff. Is randomness "native" to computer science? In *Current Trends in Computer Science, World Scientific*, volume 2, pages 141–180, 2004. arxiv.org:0801.0289v1.

[125] Kenichi Fujibayashi, Rizal Hariadi, Sung Ha Park, Erik Winfree, and Satoshi Murata. Toward reliable algorithmic self-assembly of dna tiles: A fixed-width cellular automaton pattern. *Nano Letters*, 8(17):1791–1797, 2008.

[126] Kenichi Fujibayashi, David Yu Zhang, Erik Winfree, and Satoshi Murata. Error suppression mechanisms for dna tile self-assembly and their simulation. *Natural Computing*, 8(3):589–612, 2009.

[127] Péter Gács. Reliable computation with cellular automata. *Journal of Computer and System Sciences*, 32(1):15–78, 1986.

[128] Péter Gács. Self-correcting two-dimensional arrays. *Advances in Computing Research*, 5(223-326), 1989.

[129] Péter Gács. Reliable cellular automata with self-organization. *J. of Stat. Phys.*, 103(1-2):45–267, 2001.

[130] Péter Gács and John H. Reif. A simple three-dimensional real-time reliable cellular array. *J. Comput. Syst. Sci.*, 36(2):125–147, 1988.

[131] J. Gao, L. J. Guibas, and A. Nguyen. Deformable spanners and applications. In *Proc. of ACM Symp. on Computational Geometry (SoCG)*, pages 190–199, 2004.

[132] M. Gardner. Mathematical games: The fantastic combinations of John Conway's new solitaire game 'Life'. *Scientific American*, 223(4):120–123, 1970.

[133] Cyril Gavoille and David Peleg. Compact and localized distributed data structures. *J. of Distributed Computing*, 16(2-3):111–120, 2003. Special issue for the 20 years of the ACM Symp. on Principles of Distributed Computing (PODC).

[134] George Giakkoupis and Nicolas Schabanel. Optimal path search in small worlds: Dimension matters. In *Proc of ACM Symp. on Theory of Computing (STOC)*, volume 43, pages 393–402, 2011.

[135] Jean-Louis Giavitto. Le calcul, une notion difficile à attraper. *)i(interstices*, 2009. http://interstices.info/calcul-notion.

[136] Gnutella peer-to-peer network. http://en.wikipedia.org/wiki/Gnutella.

[137] Kurt Gödel. Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I. *Monatschefte für Mathematik und Physic*, 38:173–198, 1931. Réimprimé dans *Kurt Gödel: Collected works*, Oxford university press, Oxford, **1** (1986), 144-195.

[138] Oded Goldreich. Randomized methods in computation, Lecture 5: Approximate counting and uniform generation, Spring 2001. Notes de cours prises par O. Lachish, E. Ofek et U. Wieder, disponibles à http://www.wisdom.weizmann.ac.il/∼oded/.

[139] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *J. ACM*, 33(4):792–807, 1986. Version préliminaire publiée à FOCS 1984.

[140] Oded Goldreich and Leonid A. Levin. A hard-core predicate for all one-way functions. In *Proc. of ACM Symp. on Theory of Computing (STOC)*, pages 25–32, 1989.

[141] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge compexity of interactive proof systems. *SIAM Journal of Computing*, 18(1):186–208, 1989.

[142] E. Goles and S. Martinez. *Neural and automata networks, dynamical behavior and applications*, volume 58 of *Maths and Applications*. Kluwer Acad. Publish., 1990.

[143] Eric Goles, Matthieu Latapy, Clémence Magnien, Michel Morvan, and Thi Ha Duong Phan. Sandpile models and lattices: A comprehensive survey. *Theoretical Computer Science*, 322:383–407, 2004.

[144] Ronald L. Graham. Bounds for certain multiprocessing anomalies. *Bell Systems Technical J.*, 45:1563–181, 1966.

[145] Ronald L. Graham. Bounds on multiprocessing timing anomalies. *SIAM J. Appl. Math.*, 17:416–429, 1969.

[146] Ronald L. Graham. Bounds on multiprocessing anomalies and related packing algorithms. In *Proc. of AFIPS Conf.*, volume 40, pages 205–217, 1972.

[147] Ronald L. Graham. The combinatorial mathematics of scheduling. *Scientific American*, 238:124–132, 1978.

[148] Ronald L. Graham, Eugene L. Lawler, Jan K. Lenstra, and Alexander H. G. Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: a survey. *Ann. Discrete Math.*, 4:287–326, 1979.

[149] Janko Gravner and Alexander E. Holroyd. Slow convergence in bootstrap percolation. *Annals of Applied Prob.*, 18(3):909–928, 2008. http://arxiv.org/pdf/0705.1347.

[150] Lawrence F. Gray. A reader's guide to Gács's 'positive rates' paper. *J. of Stat. Phys.*, 103(1-2):1–44, 2001.

[151] David Griffeath and Cristopher Moore. Life without death is P-complete. *Complex systems*, 10:437–447, 1996.

[152] Geoffrey Grimmett. *Percolation*, volume 321 of *Grund Lehren der mathematischen Wissenschaftenehren*. Springer-Verlag, 1999. 2nd edition.

[153] Geoffrey Grimmett and David Stirzaker. *Probability and Random Process*. Oxford University Press, 3rd edition, 2001.

[154] Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer, 1988.

[155] Anupam Gupta, Robert Krauthgamer, and James R. Lee. Bounded geometries, fractals, and low-distortion embeddings. In *Proc. of IEEE Conf. on Foundations of Computer Science (FOCS)*, pages 534–543, 2003.

[156] Jacques Hadamard. Les surfaces à courbures opposées et leurs lignes géodésiques. *Journal des Mathématiques pures et appliquées*, 4:27–73, 1898.

[157] Yijie Han. Deterministic sorting in $o(n \log\log n)$ time and linear space. *J. of Algorithms*, 50:96–105, 2004.

[158] S. Har-Paled and M. Mendel. Fast construction of nets in low dimensional metrics and their applications. In *Proc. of ACM Symp. on Computational Geometry (SoCG)*, page ??????????, 2005.

[159] Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM J. of Computing*, 1999.

[160] O. J. Heilmann and E. H. Lieb. Theory of monomer-dimer systems. *Comm. in Mathematical Physics*, 25:190–232, 1972.

[161] Malte Henkel. Sur la solution de Sundman du problème des trois corps. *Philosophia Scientiæ*, 2:161–184, 2001.

[162] M. Hénon. A family of periodic solutions to the planar three-body problem, and their stability. *Celestial Mechanics*, 13:267–285, 1976.

[163] David Hilbert. Hilbert's problems, 1900. Paris, France. http://en.wikipedia.org/wiki/Hilbert's_problems.

[164] Alexander E. Holroyd. Sharp metastability threshold for two-dimensional bootstrap percolation. *Probability Theory and Related Fields*, 125(2):195–224, 2003. http://arxiv.org/pdf/math/0206132.

[165] John J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proc. Nat. Acad. Sci.*, 79(8):2554–2558, 1982.

[166] Sungjin Im and Benjamin Moseley. An online scalable algorithm for average flow time in broadcast scheduling. In *Proc. of ACM/SIAM Symp. On Discrete Algorithms (SODA)*, 2010. To be published.

[167] Russell Impagliazzo, Valentine Kabanets, and Avi Wigderson. In search of an easy witness: Exponential time vs. probabilistic polynomial time. In *Proc. of IEEE Conf. on Computational Complexity*, pages 2–12, 2001.

[168] Russell Impagliazzo, Leonid A. Levin, and Michael Luby. Pseudo-random generation from one-way functions. In *Proc. of ACM Symp. on Theory of Computing (STOC)*, pages 12–24, 1989.

[169] Russell Impagliazzo and Avi Wigderson. P = BPP if E requires exponential circuits: Derandomizing the XOR lemma. In *Proc. of ACM Symp. on Theory of Computing (STOC)*, pages 220–229, 1997.

[170] Russell Impagliazzo and Avi Wigderson. Randomness vs time: Derandomization under a uniform assumption. *J. of Computer and System Sciences*, 63(4):672–688, 2001. Version préliminaire publiée à FOCS 1998.

[171] Piotr Indyk. Algorithmic applications of low-distortion geometric embeddings (Invited tutorial). In *Proc. of IEEE Conf. on Foundations of Computer Science (FOCS)*, pages 10–33, 2001.

[172] E. Ising. Beitrag zur theorie des ferromagnetismus. *Zeitschrift für Physik*, 31:253–258, 1925.

[173] Svante Janson, Tomasz Łuczak, and Andrzej Ruciński. *Random Graphs*. John Wiley & Sons, New York, 2000.

[174] Mark Jerrum and Alistair Sinclair. Approximating the permanent. *SIAM Journal of Computing*, 18:1149–1178, 1989.

[175] Mark Jerrum, Alistair Sinclair, and Eric Vigoda. A polynomial-time approximation algorithm for the permanent of a matrix with nonnegative entries. *Journal of the ACM*, 51(4):671–697, 2004.

[176] Mark Jerrum, Leslie G. Valiant, and Vijay V. Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theoretical Computer Science*, 43:169–188, 1986.

[177] David S. Johnson. Fast algorithms for bin packing. *J. Comput. Syst. Sci.*, 8(3):272–314, 1974.

[178] David S. Johnson, Alan J. Demers, Jeffrey D. Ullman, M. R. Garey, and Ronald L. Graham. Worst-case performance bounds for simple one-dimensional packing algorithms. *SIAM J. Comput.*, 3(4):299–325, 1974.

[179] Valentine Kabanets and Russell Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. In *Proc. of ACM Symp. on Theory of Computing (STOC)*, pages 355–364, 2003.

[180] Bala Kalyanasundaram and Kirk Pruhs. Speed is as powerful as clairvoyance. *J. of the ACM*, 47:214–221, 2000. An abstract version was published in FOCS1995.

[181] Bala Kalyanasundaram and Kirk Pruhs. Minimizing flow time nonclairvoyantly. *J. ACM*, 50(4):551–567, 2003. An abstract version was published in FOCS1997.

[182] Bala Kalyanasundaram, Kirk Pruhs, and M. Velauthapillai. Scheduling broadcasts in wireless networks. In *Proc. of European Symp. on Algorithms*, pages 290–301, 2000.

[183] František Kardoš, Daniel Král', Jozef Miškuf, and Jean-Sébastien Sereni. Fullerene graphs have exponentially many perfect matchings. In *J. of Mathematical Chemistry*, 2008. À paraître. http://kam.mff.cuni.cz/∼sereni/.

[184] Alan Kirman. Economies as complex interactive systems: Of ants and men. In *Winter School on Complex Systems*, Lyon, France, 2002.

[185] Alan Kirman. *Complexity and the Economy: Implications for Economic Policy*, chapter Individual and Aggregate Behaviour: Of Ants and Men. Edward Elgar Publishing, 2005.

[186] Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632, 1999.

[187] Jon M. Kleinberg. Navigation in a small world. *Nature*, 406:845, 2000.

[188] Jon M. Kleinberg. The small-world phenomenon: an algorithmic perspective. In *Proc. of ACM Symp. on Theory of Computing (STOC)*, pages 163–170, 2000.

[189] Jon M. Kleinberg. The small-world phenomenon and decentralized search. *SIAM News*, 37(3), 2004.

[190] Jon M. Kleinberg. Complex networks and decentralized search algorithms. In *Proc. of the Int. Congress of Mathematicians (ICM)*, pages 1–26, 2006. Nevanlinna prize recipient presentation.

[191] Andrei N. Kolmogorov. Trois approches à la définition du concept de quantité d'information. *Probl. Peredachi Inform.*, 1:3–11, 1965.

[192] R. Kumar, Prabhakar Raghavan, S. Rajagopalan, D. Sivakumar, A. Tomkins, and E. Upfal. Stochastic models for the web graph. In *Proc. of Symp. on Foundations of Computer Science (FOCS)*, pages 57–65, 2000.

[193] Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, and Andrew S. Tomkins. Recommendation systems: a probalistic analysis. In *Proc. of Symp. on Foundations of Computer Science (FOCS)*, pages 664–673, 1998.

[194] Emmanuelle Lebhar. *Algorithmes de routage et modèles aléatoires pour les graphes petits mondes*. PhD thesis, École normale supérieure de Lyon, Dec. 2005. Ref. 05ENSL0341.

[195] Emmanuelle Lebhar and Nicolas Schabanel. Almost optimal decentralized routing in long-range contact networks. In *Proc. of the Int. Colloquium on Automata, Languages and Programming (ICALP)*, volume LNCS 3142, pages 894–905, July 2004.

[196] Emmanuelle Lebhar and Nicolas Schabanel. Close to optimal decentralized routing in long-range contact networks. *Theoretical Computer Science special issue on ICALP 2004*, 348(294-310), 2005.

[197] Emmanuelle Lebhar and Nicolas Schabanel. Graph augmentation via metrics embeddings. In *Proc. of 12th Int. Conf. on Principles of Distributed Systems (OPODIS)*, volume LNCS 5401, pages 217–225, Dec. 2008.

[198] Joseph Y-T. Leung. *Handbook of scheduling: algorithms, models and performance analysis*. Chapman & Hall/CRC, 2004.

[199] Thomas M. Liggett. *Interacting particles systems*, volume 276 of *Grund. für Math. Wissen.* Springer, 1985.

[200] Thomas M. Liggett. *Stochastic Interacting Systems: Contact, Voter and Exclusion Processes*, volume 324 of *Grund. für Math. Wissen.* Springer, 1999.

[201] Nathan Linial, Eran London, and Yuri Rabinovich. The geometry of graphs and some of its algorithmic applications. *Combinatorica*, 15:215–245, 1995.

[202] Edward N. Lorenz. Deterministic non-periodic flow. *Journal of the Atmospheric Sciences*, 20(2):130–141, 1963.

[203] Benoît Mandelbrot. *Les objets fractals*. Flamarion, 1975. Version anglaise: *The fractal geometry of nature*, Freeman, San Franscico, 1977.

[204] Gurmeet Singh Manku, Moni Naor, and Udi Wieder. Know thy neighbor's neighbor: the power of lookahead in randomized P2P networks. In *Proc. of ACM Symp. on Theory of Computing (STOC)*, 2004.

[205] Irène Marcovici. Probabilistic cellular automata and their invariant measures, an analytical and algorithmic approach. Master's thesis, École normale supérieure de Lyon, 2009.

[206] Charles Martel and Van Nguyen. Analyzing kleinberg's (and other) small-world models. In *Proc. of ACM Symp. on Principles of Distributed Computing (PODC)*, pages 179–188, 2004.

[207] Charles Martel and Van Nguyen. Analyzing and characterizing small-world graphs. In *Proc. of ACM/SIAM Symp. On Discrete Algorithms (SODA)*, pages 311–320, 2005.

[208] Per Martin-Löf. The definition of random sequences. *Information and Control*, 9:602–619, 1966.

[209] Heiner Marxen. Busy beaver home page. http://www.drb.insel.de/~heiner/BB/.

[210] Claire Mathieu (Kenyon) and Nicolas Schabanel. The data broadcast problem with non-uniform transmission times. In *Proc. of ACM/SIAM Symp. On Discrete Algorithms (SODA)*, volume 10, pages 547–556, Jan. 1999.

[211] Claire Mathieu (Kenyon) and Nicolas Schabanel. The data broadcast problem with non-uniform transmission times. *Algorithmica*, 35:146–175, 2003.

[212] Claire Mathieu (Kenyon), Nicolas Schabanel, and Neal E. Young. Broadcasting data to minimize the average response time. In *Proc. of the Symp. on Theory of Computing (STOC)*, volume 32, pages 659–666, May 2000.

[213] Jacques Mazoyer. A six states minimal time solution to the firing squad synchronization problem. *Theoretical Computer Science*, 50(2):183–238, 1987.

[214] Barry M. McCoy and Tai Tsun Wu. *The Two-Dimensional Ising Model*. Harvard University Press, 1974.

[215] Milena Mihail and Christos H. Papadimitriou. On the eigenvalue power law. In *Proc. of RANDOM*, pages 254–262, 2002.

[216] Stanley Milgram. The small world problem. *Psych. Today*, 2:60–67, 1967.

[217] H. Minc. *Encyclopedia of mathematics and its applications*, volume 6, chapter Permanents. Addison-Wesley, 1982.

[218] M. Minsky. *Computation: Finite and infinite machines*. Prentice Hall, Englewood Cliffs, 1967.

[219] Cristopher Moore. Braids in classical gravity. *Physical Review Letters*, 70:3675–3679, 1993.

[220] Cristopher Moore and Michael Nauenberg. New periodic orbits for the n-body problem, 2005. arXiv:math/0511219v1.

[221] J. L. Moreno. *Who shall survive?* Beacon House, Beacon, NY, 1934.

[222] R. Morris. Counting large numbers of events in small registers. *Communications of the ACM*, 21(10):840–842, 1977.

[223] Keith A. Mott and David Peak. Stomatal patchiness and task-performing networks. *Annals of Botany*, 99:219–226, 2007. Clips available at http://aob.oxfordjournals.org.

[224] Rajeev Motwani, Steven Philipps, and Eric Torng. Non-clairvoyant scheduling. *Theoretical Computer Science (Special issue on dynamic and on-line algorithms)*, 130:17–47, 1994.

[225] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge university press, 1995.

[226] Frank D. Murgolo. Anomalous behavior in bin packing algorithms. *Discrete Applied Mathematics*, 21:229–243, 1988.

[227] N. Nauenberg. Periodic orbits for the three particles with finite angular momentum. *Physics Letters*, 292:93–99, 2001.

[228] Mark E. J. Newman. The structure and function of complex networks. *SIAM Review*, 45(2):167–256, 2003.

[229] Noam Nisan and Avi Wigderson. Hardness vs. randomness. *J. of Computer Systems and Sciences*, 49(2):149–167, 1994. Version préliminaire publiée à FOCS 1988.

[230] Nicolas Ollinger. *Automates celullaires: structures*. PhD thesis, École normale supérieure de Lyon, 2002.

[231] Nicolas Ollinger. Universalities in cellular automata; a (short) survey. In *Proc. des Journées Automates Cellulaires (JAC)*, volume 1, pages 102–118, Uzès, France, 2008.

[232] Lars Onsager. Crystal Statistics. I. A two-dimensional model with an order-disorder transition. *Phys. Rev.*, 65:117–149, 1944.

[233] Opengl. `http://www.opengl.org/`.

[234] Christos H. Papadimitriou, Prabhakar Raghavan, H. Tamaki, and S. Vempala. Latent semantic indexing: a probabilistic analysis. In *Proc. of ACM Symp. on Principles of Database Systems (PODS)*, pages 159–168, 1998.

[235] Romualdo Pastor-Satorras and Alessandro Vespignani. Epidemic spreading in scale-free networks. *Physical Review Letters*, 86:3200–3203, 2001.

[236] David Peak, Jevin D. West, Susanna M. Messinger, and Keith A. Mott. Evidence for complex, collective dynamics and emergent, distributed computation in plants. In *Proc. of the Nat. Academy of Science of the USA (PNAS)*, volume 101, pages 918–922, 2004. Clips available at `http://bioweb.usu.edu/kmott/Complexity_Web_Page/complexity_homepage.htm`.

[237] Rudolf Peierls. On the Ising model of ferromagnetism. *Proc. Camb. Phil. Soc.*, 1936. Available in *"Selected scientific papers of Sir Rudolf Peierls: with commentary"* (Richard H. Dalitz and Rudolf Peierls eds).

[238] Cynthia A. Phillips, Cliff Stein, Eric Torng, and Joel Wein. Optimal time-critical scheduling via resource augmentation. In *Proc. of ACM Symp. on Theory of Computing (STOC)*, pages 140–149, 1997.

[239] Henri Poincaré. Sur le problème des trois corps et les équations de la dynamique. *Acta Mathematica*, 13:1–270, 1890.

[240] Claude Ponti. *Le Nakakoué*. l'école des loisirs, 1997.

[241] E. Post. Formal reductions of the general combinatorial decision problem. *American Journal of Mathematics*, 65(2):197–215, 1943.

[242] Alain Prochiantz. Processus morphogénétiques. Cours du collège de France, 2007-2008.

[243] James G. Propp and David B. Wilson. Exact sampling with coupled Markov chains and applications to statistical mechanics. *Random Structures and Algorithms*, 9(1-2):223–252, 1996.

[244] Kirk Pruhs. Competitive online scheduling for server systems. *SIGMETRICS Performance Evaluation Review*, 34(4):52–58, 2007.

[245] Kirk Pruhs, Jiri Sgall, and Eric Torng. *Online Scheduling*, chapter 15 of *Handbook of scheduling: Algorithms, models and performance analysis* (Joseph Y-T. Leung ed. [198]). Chapman & Hall/CRC, 2004.

[246] Kirk Pruhs and Patchrawat Uthaisombut. A comparison of multicast pull models. In *Proc. of European Symp. on Algorithms (ESA)*, pages 808–819, 2002.

[247] Dana Randall. Rapidly mixing Markov chains with applications in computer science and physics. *IEEE Computing in Science and Engineering*, pages 30–41, Mars-Avril 2006.

[248] Dana Randall. Slow mixing of glauber dynamics via topological obstructions. In *Proc. of ACM/SIAM Symp. On Discrete Algorithms (SODA)*, pages 870–879, 2006.

[249] Iván Rapaport. *Inducing an order on cellular automata by a grouping operation*. PhD thesis, École normale supérieure de Lyon, 1997.

[250] S. Ratsanamy, P. Francis, M. Handley, Richard Karp, and S. Shenker. A scalable content-addressable network. In *Proc. of ACM Special Interest Group on Data Communications (SIGCOMM)*, pages 161–170, 2001.

[251] Damien Regnault. Abrupt behavior changes in cellular automata under asynchronous dynamics. In *Proc. of European Conference on Complex Systems (ECCS)*, pages 116–123, 2006.

[252] Damien Regnault. Directed percolation arising in stochastic cellular automata analysis. In *Proc. of Symp. on Mathematical Foundation of Computer Science (MFCS)*, volume LNCS 5162, pages 563–574, 2008.

[253] Damien Regnault. Quick energy drop in stochastic 2D Minority. In *Proc. of Int. Conf. on Cellular Automata for Reseach and Industry (ACRI)*, volume LNCS 5191, pages 307–314, 2008.

[254] Damien Regnault. *Sur les automates cellulaires probabilites: comportements asynchrones*. PhD thesis, École normale supérieure de Lyon, Nov. 2008. Ref. 07ENSL0494.

[255] Damien Regnault, Nicolas Schabanel, and Éric Thierry. Progresses in the analysis of stochastic 2D cellular automata: a study of asynchronous 2D minority. In *Proc. of Symp. on Mathematical Foundation of Computer Science (MFCS)*, volume LNCS 4708, pages 320–332, Aug. 2007.

[256] Damien Regnault, Nicolas Schabanel, and Éric Thierry. On the analysis of "simple" 2d stochastic cellular automata. In *Proc. of Int. Conf. on Language and Automata Theory and Applications (LATA)*, volume LNCS 5196, pages 452–463, Mar. 2008.

[257] Damien Regnault, Nicolas Schabanel, and Éric Thierry. On the analysis of "simple" 2D stochastic cellular automata. *Invited publication to the special issue of Discrete Mathematics and Theoretical Computer Science in honor of Philippe Flajolet*, 12(2):263–294, 2009.

[258] Damien Regnault, Nicolas Schabanel, and Éric Thierry. Progresses in the analysis of stochastic 2D cellular automata: a study of asynchronous 2D minority. *Theoretical Computer Science*, 410(47-49):4844–4855, 2009. doi:10.1016/j.tcs.2009.06.024.

[259] Paul Rendell. Webpage: This is a Turing machine implemented in Conway's game of life, 2005. http://rendell-attic.org/gol/tm.htm.

[260] Henry Gordon Rice. Classes of recursively enumerable sets and their decision problems. *Trans. Amer. Math. Soc.*, 74:358–366, 1953.

[261] Gaëtan Richard. Rule 110 : Universality and catenations. In *Proc. des Journées Automates Cellulaires (JAC)*, volume 1, pages 141–160, 2008.

[262] Ron Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978. http://en.wikipedia.org/wiki/RSA.

[263] Julien Robert. *Ordonnancement non-clairvoyant avec contraintes de dépendances*. PhD thesis, École normale supérieure de Lyon, Dec. 2009.

[264] Julien Robert and Nicolas Schabanel. Non-clairvoyant batch set scheduling: Fairness is fair enough. In *Proc. of European Symp. on Algorithms (ESA)*, volume LNCS 4698, pages 742–753, Oct. 2007.

[265] Julien Robert and Nicolas Schabanel. Pull-based data broadcast with dependencies: Be fair to users, not to items. In *Proc. of ACM/SIAM Symp. On Discrete Algorithms (SODA)*, pages 238–247, Jan. 2007.

[266] Julien Robert and Nicolas Schabanel. Non-clairvoyant scheduling with precedence constraints. In *Proc. of ACM/SIAM Symp. On Discrete Algorithms (SODA)*, pages 491–500, Jan. 2008.

[267] Julien Robert and Nicolas Schabanel. Ordonnancement non-clair voyant avec dépendances : analyse de **Laps**$_\beta$ ∘ **Equi**. In *Proc. of Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications (Algotel)*, 2009. (4 pages en français) Distributed on-line: http://hal.archives-ouvertes.fr/hal-00383347/en/.

[268] Julien Robert and Nicolas Schabanel. Ordonnancement non-clairvoyant: petites simplifications et améliorations de l'analyse de la famille d'algorithmes **Laps**$_\beta$. In *Proc. of Rencontres Francophones sur les Aspects Algorithmiques des Télécommunications (Algotel)*, 2009. (4 pages en français)
Distributed on-line: http://hal.archives-ouvertes.fr/hal-00384663/en/.

[269] Neil Robertson and Paul D. Seymour. Graph minors. ii. algorithmic aspects of tree-width. *J. Algorithms*, 7(3):309–322, 1986.

[270] Neil Robertson and Paul D. Seymour. Graph minors. xx. wagner's conjecture. *J. Comb. Theory*, 92(2):325–357, 2004.

[271] Sara Robinson. Still guarding secrets after years of attacks, RSA earns accolades for its founders. *SIAM News*, 36(5), June 2003. http://en.wikipedia.org/wiki/RSA.

[272] Paul W. K. Rothemund. Folding DNA to create nanoscale shapes and patterns. *Nature*, 440:297–302, 2006.

[273] Paul W. K. Rothemund and Erik Winfree. The program-size complexity of self-assembled squares (extended abstract). In *Proc. of ACM Symp. on Theory of Computing (STOC)*, pages 459–468, 2000.

[274] Paul W.K. Rothemund. *Theory and Experiments in Algorithmic Self-Assembly*. PhD thesis, University of Southern California, 2001.

[275] Paul W.K. Rothemund, Nick Papadakis, and Erik Winfree. Algorithmic self-assembly of DNA Sierpinski triangles. *PLoS Biology*, 2(12):2041–2053, 2004.

[276] David Ruelle. *Hasard et Chaos*. Odile Jacob, 1991.

[277] Nicolas Schabanel. *Algorithmes d'approximation pour les télécommunications sans fil : Ordonnancement pour la dissémination de données et Allocation statique de fréquences*. PhD thesis, École normale supérieure de Lyon, 2000. Réf. 00ENSL0145.

[278] Nicolas Schabanel. The databroadcast problem with preemption. In *Proc. of Int. Symp. on Theoretical Aspects of Computer Science (STACS)*, volume LNCS1770, pages 181–192, Feb. 2000.

[279] Nicolas Schabanel. Thèse idiote: le hasard fabrique des certitudes. *Le Minotaure*, 1:66–69, Avril–juin 2003.

[280] Nicolas Schabanel. Asynchronous randomized automata: how does randomness affect decentralized algorithms execution?'. Conférence invitée au séminaire « Approximation and randomized algorithms », Dagstuhl seminar 05201, Juin 2005.

[281] Nicolas Schabanel. Perspectives on small world: Will artificial models explain real networks? Conférence-débat au bilan à mi-parcours de l'action européenne COST 295 DYNAMO, Lisbonne, Juil. 2007.

[282] Nicolas Schabanel. On stochastic cellular automata. Conférence invitée pour le 60ème anniversaire de Philippe Flajolet à l'École normale supérieure, Paris, Déc. 2008.

[283] Nicolas Schabanel. Asynchronous outer-totalistic cellular automaton simulator, 2009. http://www.liafa.jussieu.fr/~nschaban/OT-Simulator/.

[284] Thomas C. Schelling. Dynamic models of segregation. *Journal of Mathematical Sociology*, 1(1):119–132, 1971.

[285] Thomas C. Schelling. *Micromotives and macrobehavior*. New York: Norton, 1978.

[286] B. Schönfisch and A. M. de Roos. Synchronous and asynchronous updating in cellular automata. *BioSystems*, 51:123–143, 1999.

[287] A. Shamir. On the generation of cryptographically strong pseudorandom sequences. *ACM Trans. on Computer Sys.*, 1(1):38–44, 1983. Version préliminaire publiée à ICALP 1981.

[288] Claude Shannon. A mathematical theory of communication. *Bell System Tech. J.*, 27:379–423, 623–656, 1948.

[289] Claude Elwood Shanon. Communication theory of secrecy systems. *Bell System Tech. J.*, 28(1):59–98, 1949.

[290] H. A. Simon. On a class of skew distribution functions. *Biometrika*, 42(3):425–440, 1955.

[291] Google SketchUp. Téléchargeable gratuitement à `http://sketchup.google.com/`.

[292] Daniel D. Sleator and Robert E. Tarjan. Amortized efficiency of list update and paging rules. *Commun. ACM*, 28(2):202–208, 1985.

[293] Aleksandrs Slivkins. Distance estimation and object location via rings of neighbors. In *Proc. of ACM Symp. on Principles of Distributed Computing (PODC)*, pages 41–50, 2005.

[294] W. E. Smith. Various optimizers for single-state production. *Naval Res. Logis. Quart.*, 3:56–66, 1965.

[295] Ray Solomonoff. A formal theory of inductive inference. *Information and Control*, 7:1–22, 1965.

[296] David Soloveichik and Erik Winfree. Complexity of self-assembled shapes. *SIAM J. of Computing*, 36(6):1544–1569, 2004.
`http://www.dna.caltech.edu/DNAresearch_publications.html`.

[297] Ion Stoica, Robert Morris, David R. Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord : a scalable peer-to-peer lookup service for internet applications. In *Proc. of ACM Special Interest Group on Data Communications (SIGCOMM)*, pages 149–160, 2001.

[298] K. F. Sundman. Mémoire sur le problème à trois corps. *Acta Mathematica*, 36(105-179), 1913.

[299] Andrew S. Tanenbaum and Robbert van Renesse. Distributed operating systems. *ACM Comput. Surv.*, 17(4):419–470, 1985.

[300] TED Ideas worth spreading. `http://www.ted.com/`.

[301] Philippe Testard-Vaillant. Les promesses tenues des nanos. *Journal du CNRS*, 237, 2009.

[302] Guillaume Theyssier. *Automates cellulaires: un modèle de complexités*. PhD thesis, École normale supérieure de Lyon, décembre 2005.

[303] Mikkel Thorup and Uri Zwick. Approximate distance oracles. *J. ACM*, 52(1):1–24, 2005.

[304] André L. Toom. Stable and attractive trajectories in multicomponent systems. *Advances in Probability and Related Topics*, 6:549–576, 1980.

[305] Pablo Triana. Les recettes apparemment simples de la réussite. *Le Monde*, 14 décembre 2007.

[306] Alan M. Turing. On computable numbers with an application to the entscheidungs problem. In *Proc. of the London Mathematical Society*, volume 42, pages 230–265, 1936.

[307] Christopher Umans. Pseudo-random generators for all hardnesses. *J. of Computer and System Sciences*, 67(2):419–440, 2003.

[308] Vijay V. Vazirani. *Approximation Algorithms*. Springer, 2001.

[309] Vijay V. Vazirani. *Algorithmes d'approximation*. Springer, Collection Iris, 2006. Traduction de Nicolas Schabanel, 445 pages, paru en mars 2006.

[310] J. von Neumann. Zur Theorie des Gesellschaftsspiele. *Math. Ann.*, 100:295–320, 1928.

[311] Abraham Waksman. An optimum solution to the firing squad synchronization problem. *Information and Control*, 9(1):66–78, 1966.

[312] Q. D. Wang. The global solution of the n-body problem. *Celestial Mechanics and Mechanical Astronomy*, 50:73–88, 1991.

[313] Hal Wasserman and Manuel Blum. Software reliability via run-time result-checking. *Journal of the ACM*, 44(6):826–849, 1997.

[314] J. Watson and F. Crick. Molecular structure of nucleic acids; a structure for deoxyribose nucleic acid. *Nature*, 171(4356):737–8, 1953.

[315] Ducan J. Watts and Steven H. Strogatz. Collective dynamics of "small-world" networks. *Nature*, 393(440-442), 1998.

[316] Wikipedia. Antiferromagnetism. http://en.wikipedia.org/wiki/Antiferromagnetism.

[317] Wikipedia. Informatics. http://en.wikipedia.org/wiki/Informatics.

[318] Wikipedia. Minesweeper. http://en.wikipedia.org/wiki/Minesweeper_(computer_game).

[319] Wikipedia. P-complete problems. http://en.wikipedia.org/wiki/P-complete.

[320] Wikipedia. Paterson's worms. http://en.wikipedia.org/wiki/Paterson's_worms.

[321] Erik Winfree. *Algorithmic Self-Assembly of DNA*. PhD thesis, Caltech, 1998.

[322] Erik Winfree, Furong Liu, Lisa Wenzler, and Nadrian C. Seeman. Design and self-assembly of two-dimensional DNA crystals. *Nature*, pages 539–544, 1998.

[323] S. Wolfram. Statistical mechanics of cellular automata. *Reviews of Modern Physics*, 55:601–644, 1983.

[324] S. Wolfram. Universality and complexity in cellular automata. *Physica D*, 10:1–35, 1984.

[325] S. Wolfram. *A new kind of science*. Wolfram Media Inc., 2002.

[326] Bernard Wong, Aleksandrs Slivkins, and Emin Gün Sirer. Meridian: a lightweight network location service without virtual coordinates. In *Proc. of ACM Special Interest Group on Data Communications (SIGCOMM)*, pages 85–96, 2005.

[327] Fa-Yueh Wu. The Potts model. *Rev. Mod. Phys.*, 28:235–268, 1982.

[328] Andrew C-C. Yao. Probabilistic computations: Towards a unified measure of complexity. In *Proc. of Symp. on Foundations of Computer Science (FOCS)*, pages 222–227, 1977.

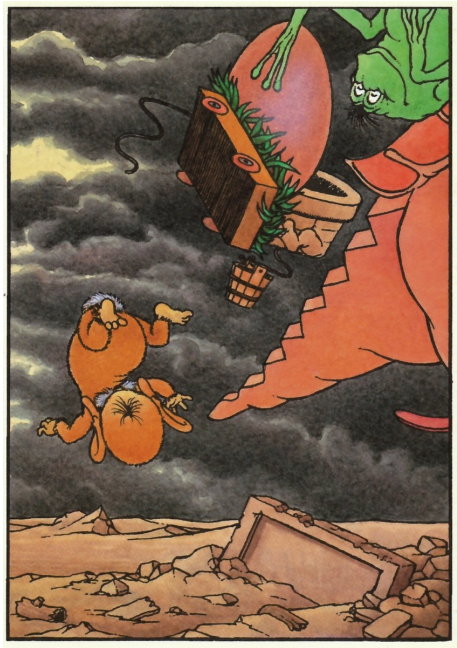
[329] Andrew C-C. Yao. New algorithms for bin packing. *J. ACM*, 27(2):207–227, 1980.

[330] Andrew C-C. Yao. Theory and applications of trapdoor functions. In *Proc. of IEEE Conf. on Foundations of Computer Science (FOCS)*, pages 80–91, 1982.

[331] Neal E. Young. *Encyclopedia of Algorithms*, chapter Online Paging and Caching. Springer, 2008.

[332] G. U. Yule. A mathematical theory of evolution, based on the conclusions of Dr. J. C. Willis. *Philos. Trans. Roy. Soc. London*, Ser. B 213:21–87, 1924.

[333] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. Kubiatowicz. Tapestry: A resilient global-scale overlay for service deployment. *IEEE J. on Selected Areas in Communications*, 22(1):41–53, 2004.



Comme Zouc ne savait pas
exactement jusqu'où il pouvait aller
trop loin, il sauta.

extrait de [240] par Claude Ponti