Thèse de Doctorat de l'Université Paris VI
Pierre et Marie Curie

Spécialité

Réseaux Informatiques

présentée par

Thomas Begin

pour obtenir le grade de

Docteur de l'université Pierre et Marie Curie

Modélisation et Calibrage Automatiques de Systèmes

Soutenue le 5 décembre 2008 devant le jury composé de

Raymond MARIE	**Rapporteur**	**Professeur, Université de Rennes 1**
Philippe NAIN	**Rapporteur**	**Directeur de Recherche à l'INRIA**
Alexandre BRANDWAJN	**Examinateur**	**Professeur, Université de Californie à Santa Cruz**
Philippe CHRETIENNE	**Examinateur**	**Professeur, Université Pierre et Marie Curie**
Bernd WOLFINGER	**Examinateur**	**Professeur, Université de Hambourg**
Serge FDIDA	**Directeur**	**Professeur, Université Pierre et Marie Curie**
Bruno BAYNAT	**Encadrant**	**MdC, Université Pierre et Marie Curie**

Thèse de Doctorat de l'Université Paris VI
Pierre et Marie Curie

Spécialité

Réseaux Informatiques

présentée par

Thomas Begin

pour obtenir le grade de

Docteur de l'université Pierre et Marie Curie

Modélisation et Calibrage Automatiques de Systèmes

Soutenue le 5 décembre 2008 devant le jury composé de

Raymond MARIE	**Rapporteur**	**Professeur, Université de Rennes 1**
Philippe NAIN	**Rapporteur**	**Directeur de Recherche à l'INRIA**
Alexandre BRANDWAJN	**Examinateur**	**Professeur, Université de Californie à Santa Cruz**
Philippe CHRETIENNE	**Examinateur**	**Professeur, Université Pierre et Marie Curie**
Bernd WOLFINGER	**Examinateur**	**Professeur, Université de Hambourg**
Serge FDIDA	**Directeur**	**Professeur, Université Pierre et Marie Curie**
Bruno BAYNAT	**Encadrant**	**MdC, Université Pierre et Marie Curie**

Remerciements

Je souhaite d'abord remercier M. Raymond Marie, Professeur à l'Université de Rennes 1, et M. Philippe Nain, Directeur de Recherche à l'INRIA, d'avoir accepté d'être les rapporteurs de ma thèse.

Je souhaite également remercier M. Alexandre Brandwajn, Professeur à l'Université de Californie Santa Cruz, M. Philippe Chrétienne, Professeur à l'Université Pierre et Marie Curie, et M. Bernd Wolfinger, Professeur à l'Université d'Hambourg, d'avoir consenti à juger et évaluer ce travail.

Je remercie M. Serge Fdida, Professeur à l'Université Pierre et Marie Curie, d'avoir dirigé cette thèse au sein du Laboratoire d'Informatique de Paris 6 et de m'avoir crédité de sa confiance au cours de ces trois années.

Je souhaite particulièrement remercier M. Bruno Baynat, Maître de Conférence à l'Université Pierre et Marie Curie, pour avoir accepté d'encadrer cette thèse, pour ses précieux conseils, pour son investissement total y compris sur les aspects les moins captivants d'un doctorat et pour ses encouragements sans relâche. Sa sympathie, sa franchise et sa rigueur ont constitué pendant ces trois années un vecteur essentiel à mes travaux de thèse, et aucun mot ne saurait traduire ma reconnaissance pour tout cela !

Je souhaite également remercier M. Alexandre Brandwajn, Professeur à l'Université de Californie Santa Cruz, pour m'avoir offert sa collaboration pendant les six mois passés en Californe. Je remercie particulièrement M. Brandwajn pour m'avoir impliqué sur ses travaux de recherche, pour son aide plus que conséquente à mes travaux de thèse et aussi, pour son amitié sans faille.

Je souhaite exprimer une pensée particulière aux membres de l'équipe « Réseaux et Performances » du Laboratoire d'Informatique de Paris 6 avec qui j'ai travaillé. Grâce à eux, les moments passés au LIP6 ont été des moments sympathiques, et nos moments de détente de franches rigolades. Je remercierai 4 fois Chloé qui à maints égards fut une compagnon de route exceptionnelle pour ces 3 années, Thomas Bourgeau avec qui toute discussion scientifique ou d'autre choix est passionnante, Fehmi pour ses astuces, Mehdi pour son humour inégalable, Xavier « le belge », Guthemberg et Benjamin pour leur amitié ainsi que tous les autres membres de l'équipe.

Je remercie également Francis Sourd, chercheur CNRS au LIP6, et Safia Kedad-Sidhoum, Maître de Conférence à l'Université Pierre et Marie Curie, pour leur soutien et leur sympathie.

Je remercie également Véronique Varenne, Marguerite Sos, Raymonde Kurinckx et Sabrina Mamert pour avoir permis de faire des aspects administratifs de mon doctorat un moment sympathique.

Je remercie enfin tous mes proches qui m'ont soutenu durant ces années de labeur. Ma famille, mes potes du PPFC, mes anciens colocs de Gare du Nord, mes amis de l'ISEP, Momo, Margit, et enfin les ouvriers et leurs perceuses au bruit démoniaque qui ont accompagné les premiers moments de rédaction de ce manuscrit à Nice !

Résumé

La pression économique qui s'exerce sur les acteurs technologiques est telle que l'évaluation de performances constitue un passage stratégique et quasi-incontournable pour les produits d'aujourd'hui. Les systèmes informatiques et les réseaux de communication modernes n'échappent pas à cette exigence. Que ce soit pour la conception, ou pour configurer et améliorer un système opérationnel, les décisions sont habituellement prises en s'appuyant sur un modèle du système. Les outils mis en oeuvre pour la modélisation du système peuvent varier selon les domaines et les besoins. Cependant, la complexité croissante, la taille grandissante et la connaissance parfois limitée des systèmes informatiques et de communications rendent cette opération difficile, si ce n'est impossible dans certains cas.

Ce rapport de thèse s'intéresse au calibrage et à la génération automatiques de modèles. Une nouvelle méthode de modélisation, nommée HLM (« High Level Modeling ») est proposée afin de permettre la modélisation rapide et automatique de systèmes opérationnels de type « boîte noire » pour lesquels on dispose uniquement de mesures. Le principe de la méthode HLM repose sur une approche originale qui consiste à rechercher parmi un ensemble présupposé de modèles génériques, si une fois correctement calibré, l'un de ces modèles permet de reproduire le comportement d'entrée/sortie du système considéré tel qu'il est décrit par les mesures. Ce rapport de thèse comprend également la description d'une méthode de calibrage automatique permettant de déterminer la valeur des paramètres indéterminés d'un modèle présupposé de type file d'attente à partir d'un ensemble de mesures. La recherche du calibrage est formulée comme un problème d'optimisation numérique pour lequel un algorithme de recherche efficace est proposé.

Le premier chapitre de ce rapport de thèse définit des notions clés à la compréhension des chapitres suivants. Le second chapitre présente le fonctionnement, la mise en oeuvre ainsi que les performances de la méthode de calibrage automatique proposée qui s'adresse aux modèles de type file d'attente. Le troisième chapitre décrit la méthode de modélisation HLM en présentant les modèles génériques considérés et la façon de les calibrer aux mesures. Le quatrième chapitre montre des exemples d'applications possibles de la méthode HLM sur des systèmes réels et variés. Enfin, le cinquième chapitre de ce rapport concerne l'influence que peuvent avoir les propriétés distributionnelles d'ordre supérieur sur les performances de files classiques de la théorie des files d'attente.

Mots clés

Modélisation, Calibrage Automatique, Boîte noire, Mesures, Files d'attente, Moments d'ordre supérieur

Abstract

Economic and technological pressures make performance evaluation a strategically important step throughout the life cycle of complex systems. This is obviously true for computer systems and recent communication networks. Their design as well as their configuration, tuning or dimensioning can derive important benefits from the use of models. Techniques and tools used to model a system may vary depending on the type of system and objectives of the model. However, the growing complexity, the increasing size and the possibly limited knowledge of computer systems and communication networks tend to make this modeling much harder to perform, if not intractable.

This thesis considers the problem of automatic calibration and generation of models. A novel modeling approach, referred as to HLM ("High Level Modeling"), is proposed to model quickly and automatically existing systems. Such systems are seen as a "black box", known mainly through measurements. In its essence, the HLM approach considers a predefined set of generic models, and attempts to discover if any of them, when correctly calibrated, can reproduce the input/output behavior of the studied system as it is described by the measurements. This thesis includes also the description of an automatic calibration method that aims at finding the parameters values for a given queueing model based on a set of measurements. The search for the calibration is set as a numerical optimization problem and an ad hoc and efficient search algorithm is proposed to solve it.

The first chapter of this thesis defines some key notions used in the following chapters. The second chapter describes the automatic calibration method, as well as its implementation and its performance. The third chapter presents the HLM modeling method. This includes the description of the generic models considered and the way to calibrate them to the measurements. The fourth chapter shows possible applications of the HLM method based on various real-life systems. Finally, the fifth chapter considers the influence that higher-order distributional properties may have on the performance of classical queueing theoretical systems.

Keywords

Modeling, Calibration, Automatic, Black box, Measurements, Queues, Higher-order moments

Table des matières

Avant-propos

Le plan de ce rapport a été pensé de telle manière que certains chapitres puissent être lus indépendamment les uns des autres.

Nous avons regroupé dans le Chapitre 1 les notions essentielles communes à tous les chapitres.

Le Chapitre 2 traitant de la méthode de calibrage automatique proposée peut être lu séparément des chapitres suivants.

Les Chapitres 3 et 4, qui présentent le fonctionnement de la méthode de modélisation HLM ainsi que des exemples d'applications possibles, peuvent être compris sans avoir lu au préalable le Chapitre 2.

Le Chapitre 5 qui concerne l'importance des propriétés distributionnellles d'ordre supérieur pour des modèles classiques issus de la théorie des files d'attente est partiellement décorrélé du thème initial de cette thèse, et peut donc être lu séparément.

Etant donné la diversité des sujets abordés dans ce rapport de thèse, les états de l'art relatifs à chacun des aspects abordés ne sont pas regroupés au sein d'un seul chapitre, mais distribués à l'intérieur de chacun des chapitres concernés.

Introduction

L'évaluation de performances

Le secteur des hautes technologies connaît depuis de nombreuses années une croissance forte mais, comme pour beaucoup d'autres secteurs, son marché est extrêmement concurrentiel et la compétition entre les produits y est très rude. La pression économique qui s'exerce sur les sociétés de haute technologie est telle que l'*évaluation de performances* constitue un passage quasi-incontournable dans la fabrication des produits. Les systèmes informatiques et les réseaux de communication n'échappent pas à cette exigence. Que ce soit pour la conception, ou pour configurer et améliorer un système opérationnel, les décisions prises par les sociétés et leurs laboratoires de recherche s'appuient habituellement sur une évaluation de performances. L'analyse de ces résultats peut permettre aux sociétés de s'assurer du bon fonctionnement, de l'utilisation optimale des ressources ou encore de la minimisation des coûts de leur produits.

Plusieurs méthodes permettent d'évaluer les performances d'un système. La plus naturelle d'entre elles peut-être, l'*expérimentation*, consiste à mettre en place les conditions souhaitées de fonctionnement du système, puis à tester par des expériences et des mesures le comportement du système [47]. Cette solution, simple en apparence, offre l'avantage considérable d'évaluer les performances réelles du système mais présente, en contrepartie, plusieurs inconvénients. En voici les principaux.

L'expérimentation peut se révéler être une solution à la fois onéreuse, longue, complexe à mettre en oeuvre et pouvant délivrer au final des résultats peu fiables et peu lisibles. Premièrement, lorsque le système concerné existe, cette solution implique des interventions sur le système pour le modifier, ainsi qu'une interruption de son service si le système est en exploitation. Si le système est en cours de conception, un prototype doit être fabriqué. Deuxièmement, le système doit être correctement instrumenté pour pouvoir collecter, stocker et rassembler les mesures afin de les traiter

et d'en extraire les résultats recherchés. Cette étape est décisive car elle doit garantir la pertinence des résultats, notamment en s'assurant que l'opération de mesure ne biaise pas le comportement du système et ne compromet pas ainsi l'intégrité des résultats [33, 87, 47]. Troisièmement, des conditions de fonctionnement particulières peuvent être également requises, comme par exemple la mise en place d'un autre type de charge de travail en entrée du système. La génération et le contrôle de ces conditions peuvent être rendus très difficiles, notamment si le système interagit avec un extérieur non maîtrisable [93, 36, 43]. Quatrièmement, les résultats obtenus par une expérimentation apportent généralement un éclairage très limité sur les paramètres clés du système et procurent donc peu d'intuition sur son fonctionnement [47]. Enfin, notons que l'expérimentation constitue une solution techniquement impossible lorsque les mesures doivent avoir lieu sur des parties non accessibles du système. Par exemple, dans le cas d'Internet, les opérateurs sont généralement rétifs à l'installation de sondes par un tiers sur leur réseau.

La modélisation

La modélisation constitue une alternative à l'expérimentation pour évaluer les performances d'un système. Cette solution repose sur la définition et la résolution d'un modèle. En informatique, on définit généralement un *modèle* comme une représentation théorique souvent simplifiée et relativement abstraite d'un système, capable de reproduire certains aspects de son comportement. C'est la résolution du modèle, suivie du calcul de ses performances, qui établit l'évaluation de performances du système [5, 47, 62, 43]. Ainsi, à cause des différences entre le système et le modèle, que l'on désigne par l'*erreur de modélisation* [34], la solution obtenue par le modèle constitue généralement une approximation des performances du système. Ceci étant dit, l'utilisation d'un modèle présente de précieux atouts que nous détaillons maintenant.

A de nombreux égards, la modélisation constitue la meilleure solution pour évaluer les performances d'un système. Premièrement, pour un système en exploitation, l'utilisation d'un modèle est particulièrement pratique puisqu'elle n'entraîne ni complication, ni perturbation sur le système ; et si le système est en cours de conception, la modélisation permet d'éviter le recours répété à de coûteux prototypes du système. Deuxièmement, les conditions de fonctionnement du modèle sont entièrement connues, contrôlables et répétables à volonté. Troisièmement, la résolution du modèle peut offrir un éclairage très précieux sur le fonctionnement du système, en mettant par exemple en exergue le rôle de ses paramètres clés [5].

Les raisons exactes qui motivent la modélisation d'un système sont multiples et

variées. On peut cependant les regrouper autour de trois grandes fonctions : décrire, expliquer et prévoir. Plus concrètement, en informatique, les objectifs d'une modélisation peuvent être : la conception d'un nouveau système, le dimensionnement et la configuration d'un système existant, la compréhension plus fine du fonctionnement d'un système, et enfin la prédiction du comportement d'un système opérationnel et la planification de capacité [43, 47, 93]. Pour ne citer qu'un seul exemple, l'essor rapide des réseaux CSMA comme Ethernet tient en partie au fait que des modélisations de sa couche MAC ont permis de concevoir et de calibrer des mécanismes performants, comme par exemple le « back-off » exponentiel après la détection d'une collision [30].

La modélisation repose généralement sur une *approche constructive*. Trois étapes principales interviennent dans l'élaboration d'un modèle constructif. La première étape, la *description du modèle*, est élaborée à partir des connaissances que l'on a sur le système. Le modèle recherché tend à reproduire le fonctionnement interne du système. C'est cette exigence qui caractérise l'approche constructive. La ressemblance entre le modèle et le système est souvent manifeste et certains paramètres du modèle peuvent être calqués sur ceux du système. Le *calibrage* des paramètres constitue la deuxième étape et consiste à déterminer les valeurs des paramètres du modèle. Leurs valeurs peuvent être directement obtenues à partir de la connaissance du système, ou bien décidées en fonction de mesures collectées sur le système. La dernière étape, la *résolution*, consiste comme son nom l'indique, à résoudre le modèle afin de pouvoir l'exploiter.

De nombreux formalismes existent pour décrire un modèle. Le choix du formalisme est fonction des objectifs de l'étude et du système considéré. Ainsi, les réseaux de Pétri, les machines à états et plus anciennement le GRAFCET sont habituellement employés lorsque les objectifs sont de nature qualitative. Par exemple, il peut s'agir de définir les propriétés structurelles et comportementales d'un système, de s'assurer du bon fonctionnement de synchronisations ou de vérifier l'impossibilité d'un état de blocage du système. Pour des études à visée quantitative, où c'est la recherche des performances d'un système qui prévaut, d'autres formalismes sont généralement employés. Les automates stochastiques, les modèles fluides, les modèles distributionnels, les chaînes de Markov, les files d'attente et les réseaux de files d'attente sont quelques uns des formalismes les plus souvent utilisés.

Le formalisme considéré dans ce rapport est celui des *files d'attente* [56, 5]. Les files d'attente (et les réseaux de files d'attente) sont des modèles stochastiques particulièrement adaptés à la description des phénomènes d'attente, de congestion et de saturation. Les systèmes informatiques peuvent souvent être décrits comme des assemblages de ressources matérielles ou logicielles couplés à un ensemble de demandes ou de clients qui

rivalisent pour accéder à ces ressources. Dans les ordinateurs personnels et les serveurs centraux, les ressources matérielles peuvent représenter la mémoire principale, le processeur, les disques ou les terminaux. Les ressources logicielles peuvent quant à elles correspondre aux accès bloquant en écriture à un fichier. Dans les réseaux informatiques, les ressources matérielles peuvent faire référence aux commutateurs, aux buffers ou aux liens physiques tandis qu'une ressource logicielle peut correspondre, par exemple, aux contraintes imposées par une fenêtre de contrôle d'émission sur un flot (comme le fait TCP). Lorsque les besoins en ressource des clients sont trop importants, des files d'attente se constituent, obligeant les clients à attendre avant de pouvoir être servis. Ce court raisonnement explique de façon intuitive l'utilisation courante des files d'attente pour modéliser le comportement quantitatif des systèmes informatiques [63, 2, 47, 57].

Une fois le modèle décrit à l'aide d'un formalisme et que le calibrage de ses paramètres est effectué, la résolution du modèle peut démarrer. Elle va permettre d'obtenir les grandeurs recherchées afin, par exemple, d'évaluer les performances du système associé au modèle. Dans le cadre d'une étude quantitative du modèle, les grandeurs recherchées peuvent être le débit de sortie des clients, le temps de séjour des clients dans le modèle ou encore la probabilité de rejet de clients en entrée du modèle. La résolution du modèle peut être faite de façon exacte ou approchée [34, 35], en utilisant une méthode analytique ou numérique [43] ou bien une résolution par simulation [81].

La problématique

Les obstacles à l'approche constructive

Pour pouvoir fonctionner, l'approche constructive suppose une connaissance suffisante du système étudié, et son utilisation peut se révéler complexe à chacune de ses étapes, y compris pour des spécialistes en modélisation. Premièrement, dans une approche constructive, le modèle recherché vise à reproduire les mécanismes essentiels du fonctionnement du système. Le système étudié doit donc être connu et documenté. Deuxièmement, les parties du système ayant le plus d'incidence sur son comportement doivent être finement modélisées. En revanche, on préférera agréger les aspects plus secondaires du système, ou les abstraire à un niveau élevé dans le modèle, afin de maintenir l'ensemble du modèle à la fois soluble et fidèle au système. La balance entre ces deux exigences est souvent compliquée à trouver et impose une expertise pointue du système. Une fois l'étape de description du modèle terminée, les valeurs des paramètres du modèle doivent être déterminées. Cette étape de calibrage peut poser problème lorsque les paramètres du modèle doivent être décidés à partir d'un ensemble de me-

sures collectées sur le système concerné. Les difficultés liées au calibrage sont décrites plus en détails dans la Section 2.1 du Chapitre 2. Enfin, la résolution et l'exploitation du modèle peuvent également être des étapes difficiles, nécessitant des compétences théoriques approfondies et parfois la mise en oeuvre de méthodes approchées [35].

Par ailleurs, la complexité grandissante des systèmes informatiques, liée au développement rapide des hautes technologies, tend à complexifier l'effort de modélisation. Les nouvelles générations de systèmes et de réseaux informatiques évoluent vers des produits toujours plus puissants et plus compatibles entre eux mais donc plus complexes, plus hétérogènes et de plus grande envergure.

Pour un certain nombre de systèmes, une modélisation entièrement constructive apparaît difficile, voire impossible pour quelques uns [47, 1]. Par exemple, les grands serveurs centraux, dont les dimensions peuvent être gigantesques, nécessitent un grand niveau d'abstraction et une solide expertise pour pouvoir obtenir un modèle à la fois réaliste et soluble (y compris si la résolution du modèle s'effectue par une méthode approchée ou par une simulation). Par ailleurs, un système central peut inclure des sous-systèmes, comme des contrôleurs disques, dont le fonctionnement et le comportement ne sont pas connus de l'analyste empêchant ainsi une approche constructive pour les représenter. Internet est un autre exemple d'un système dont les dimensions, la complexité, l'hétérogénéité et les parties méconnues peuvent être autant d'obstacles à une approche constructive.

Des exemples initiateurs

Prenant le contre-pied de la complexité, certains modèles (étonnamment) simples parviennent à reproduire les performances de systèmes en apparence très compliqués. Ces modèles, qui sont le plus souvent constructifs, correspondent à un niveau d'abstraction très élevé du système étudié. Ce haut niveau d'abstraction est souvent le résultat d'une grande expertise sur le fonctionnement du système. Mais comment expliquer que des modèles simples, dont les hypothèses peuvent paraître simplistes, parviennent à représenter avec suffisamment de précision des systèmes aux réalités bien plus complexes ? Nous pensons que pour de nombreux systèmes, y compris parmi les plus complexes, un nombre limité de composants ou de mécanismes du système contribue effectivement aux performances du système. Ce ou ces quelques composant(s), que seule une solide expertise permettra d'identifier, dirige(ent) l'essentiel du comportement du système. Ainsi, le goulet d'étranglement d'un système détermine souvent son seuil de saturation.

Voici quelques exemples remarquables pour lesquels la simplicité du modèle contraste avec la complexité du système concerné. On peut citer les travaux de Scherr [92] qui

a utilisé un simple modèle type « machine repairman » [2] pour représenter les performances d'un système à temps partagé. Varki et al. [101] ont eux modélisé le comportement des grappes de disques durs (« disks arrays ») à l'aide d'un réseau de files d'attente. En n'incorporant dans leur modèle que les composants et les mécanismes clés du système, Varki et al. ont pu obtenir un modèle suffisamment simple pour être résolu analytiquement, et suffisamment précis pour estimer avec justesse le débit et le temps de séjour des requêtes d'entrée/sortie (E/S) dans la grappe de disques lorsque la taille des requêtes change. Cao et al. [23] ont eux montré qu'une simple file M/G/1/K*PS [2, 56] peut suffire à représenter correctement les performances d'un serveur Apache Web à condition que ses paramètres soient correctement calibrés. En effet, une fois calibrée, la file M/G/1/K*PS reproduit avec précision les débits, les temps de séjour et les taux de perte mesurés sur le serveur Web à différents niveaux de charge. Dans le cas des réseaux, Alouf et al. [3] ont fait appel à des files à capacité finie très simples (viz. M/M/1/K, M/D/1/K) pour inférer à partir de mesures certaines propriétés du chemin entre deux hôtes. Partant de l'hypothèse que ces files peuvent suffire à modéliser certains aspects d'un lien réseau, Alouf et al. ont pu dériver des formules pour estimer la taille du buffer et l'intensité du « cross traffic » au niveau du lien goulet d'étranglement d'un chemin donné. Salamatian [90] a également proposé de modéliser certains aspects d'un chemin Internet avec une simple file d'attente et d'utiliser ce modèle pour inférer certaines propriétés sur le chemin.

La recherche d'un modèle simple, bien que pouvant apparaître parfois comme naturelle une fois le modèle décrit, peut constituer en réalité une tâche difficile. A l'expertise qu'elle exige, comme nous l'avons vu, s'ajoutent les difficultés liées au calibrage des paramètres du modèle. Du fait de son haut niveau d'abstraction, les paramètres du modèle peuvent correspondre à une agrégation de plusieurs composants, rendant leur sens obscur et difficilement associable à une réalité physique. Ainsi, bien que la résolution d'un modèle simple soit généralement assez directe, le choix du modèle et de son calibrage peuvent eux être très difficiles.

Notre axe de recherche et la genèse de notre travaux

L'idée de départ de cette thèse est que dans certains cas il serait peut être plus facile de modéliser des systèmes compliqués ou mal documentés en cherchant directement un modèle simple plutôt que d'aborder leur modélisation par une approche constructive. Ceci étant dit, il reste à décider d'une méthode pour le faire. Certes la littérature scientifique contient des exemples de modèles simples pour des systèmes complexes (nous en avons cité quelques exemples), mais à notre connaissance, très peu de travaux

se sont intéressés à élaborer une méthode complète de modélisation qui fonctionne sur ce principe.

Notre objectif initial est donc de voir si l'on peut développer une méthode de modélisation qui se fonde sur l'hypothèse de l'existence d'un modèle simple. Contrairement à l'approche constructive, le modèle recherché ne vise pas à reproduire le fonctionnement interne du système considéré mais uniquement ses performances d'entrée/sortie. Autrement dit, le système concerné peut être vu comme une boîte noire pour laquelle on dispose de mesures. La méthode de modélisation fonctionnera si elle parvient à trouver un modèle simple de type file d'attente qui reproduit les performances du système exprimées à travers des mesures, et plus généralement s'il constitue un modèle possible du système. Afin de rendre son utilisation aussi simple que possible et ainsi accessible au plus grand nombre, cette méthode doit pouvoir être entièrement automatisée.

Les mesures constituent la « matière première » de cette méthode. Elles seules déterminent la description du modèle et son calibrage. Le système étudié est perçu comme une boîte noire (i.e. son fonctionnement interne est supposé inconnu) pour laquelle on dispose simplement de mesures d'entrée/sortie. Par conséquent, aucune connaissance sur le fonctionnement interne du système, ni compétence particulière en modélisation n'est requise pour la faire fonctionner. Enfin puisqu'on suppose que l'on dispose de mesures, cette méthode se destine naturellement aux systèmes existants et opérationnels.

Notons que les objectifs de la méthode HLM peuvent être rapprochés de l'Automatique où l'on cherche à représenter le fonctionnement d'un système par une fonction de transfert sans se préoccuper si le système observé est réalisé grâce à une technologie hydraulique, mécanique ou électronique. Il s'agit par exemple de savoir s'il peut être représenté par un système linéaire du premier ou du second ordre ; les paramètres de cette fonction de transfert sont « identifiés » afin que le comportement dynamique du modèle retenu reproduise au mieux celui du système observé. Toutefois, la similitude s'arrête ici pour plusieurs raisons. Premièrement, la nature des modèles considérés dans les deux approches diffère. Dans le cas de la méthode HLM, il s'agit de modèles (hélas non linéaires) de congestion (utilisation en mode compétitif de ressources en nombre limité). Cette différence nécessite de développer des outils différents pour les calibrer, c'est-à-dire pour identifier les paramètres des modèles de la méthode HLM. Deuxièmement, la connaissance et le contrôle de la charge en entrée du système sont généralement complets dans l'Automatique. Cette propriété peut être exploitée afin de déterminer la fonction de transfert du système étudié en s'appuyant sur la mesure de ses réponses pour certaines charges (réponse impulsionnelle, etc). A l'inverse, pour les systèmes in-

formatiques considérés par la méthode HLM, les niveaux de charge qui ont engendré les performances mesurées du système peuvent être non-contrôlables, voire inconnus.

Les contributions de cette thèse

Une méthode de modélisation simple et automatique

Les contributions de cette thèse peuvent être regroupées autour de trois axes. D'abord, et c'est le sujet principal de cette thèse, nous présentons une méthode de modélisation, nommée *HLM* (pour « High Level Modeling »), dont le principe est en rupture avec l'approche constructive. La méthode permet de définir un modèle simple, de type file d'attente et correctement calibré, qui reproduit certains aspects du comportement d'entrée/sortie d'un système (ou d'un réseau) informatique. Plus précisément, le modèle produit par la méthode vise à reproduire les performances observées sur le système étudié.

La méthode HLM est une méthode de modélisation automatique et qui ne requiert que des mesures sur le système étudié pour pouvoir fonctionner. Dans sa forme initiale, la méthode ne suppose aucune information sur le fonctionnement interne du système. Le système étudié peut être vu comme une boîte noire pour laquelle on dispose uniquement de mesures d'entrée/sortie. La méthode doit pouvoir fonctionner de façon automatique ce qui permet de rendre son utilisation accessible au plus grand nombre.

Cette méthode HLM est complémentaire de l'approche constructive habituelle. Elle peut constituer dans certains cas une alternative plus performante et plus appropriée. Mais elle peut également servir de point de départ à une modélisation constructive en lui suggérant une ébauche de modèle.

La définition de la méthode HLM repose sur le principe original de rechercher un modèle simple de type file d'attente pour modéliser le comportement d'un système potentiellement très compliqué. Le fonctionnement de la méthode consiste à chercher parmi un ensemble présupposé de modèles génériques celui qui, une fois correctement calibré, permet de reproduire au mieux les performances mesurées sur le système considéré. Cet ensemble de modèles génériques inclut des modèles simples issus de la théorie des files d'attente mais également des modèles plus originaux. Ensemble, ces modèles permettent de couvrir un spectre assez large des comportements observés sur les systèmes et réseaux informatiques.

Les travaux effectués sur la méthode HLM ne se limitent pas à sa définition ; ils concernent également sa validation sur des exemples réels et la présentation de ses usages. La méthode HLM a été testée sur plusieurs situations réelles, correspondant à

des systèmes et réseaux informatiques variés. Les résultats, positifs dans leur majorité, ont débouché sur des modèles permettant de mieux prévoir, comprendre et décrire le système étudié. Premièrement, concernant la prévision, les modèles ont émis des prédictions pertinentes (vérifiées expérimentalement). Deuxièmement, certains modèles ont apporté un éclairage tantôt qualitatif, tantôt quantitatif, qui facilite la compréhension du système étudié. Troisièmement, certains modèles produits par la méthode HLM, par leur simplicité et leur propriété, permettent de guider une modélisation constructive en orientant par exemple le choix du modèle. Ces possibilités d'usage de la méthode HLM sont d'autant plus attractives que la génération des modèles est automatique et requiert uniquement des mesures.

Une méthode de calibrage automatique

Le deuxième axe des contributions concerne le calibrage automatique. Nous avons développé une méthode qui permet de calibrer automatiquement les paramètres d'un modèle de type file d'attente. Le calibrage de ses paramètres repose sur l'existence de mesures obtenues sur le système ayant occasionné le modèle. Cette méthode de calibrage permet de déterminer, en fonction des mesures, les valeurs les plus « probables » des paramètres du modèle. Des contraintes, visant à assurer la cohérence qualitative entre les mesures (du système) et le comportement du modèle, peuvent être facilement prises en compte lors de la recherche du calibrage.

Le calibrage d'un modèle est formulé comme un problème d'optimisation continue avec contraintes (non-linéaires). Deux éléments sont communs à tous les problèmes d'optimisation : la fonction objectif et l'algorithme de recherche de la solution optimale. Pour la *fonction objectif* δ, dont le rôle est d'évaluer la qualité d'un calibrage donné du modèle en quantifiant son éloignement aux points de mesure, nous avons proposé des définitions simples à mettre en oeuvre et applicables à des modèles et jeux de mesures de nature variée. Concernant, l'algorithme de recherche, nous avons développé un *algorithme type DFO* (« Derivative Free Optimization »), qui ne fait aucune hypothèse sur la fonction objectif, et a fortiori sur des propriétés du modèle. Pour fonctionner, cet algorithme ne requiert que l'évaluation de la fonction objectif δ, ce qui le rend utilisable pour la plupart des modèles, y compris ceux résolus par méthode numérique ou par simulation. Cet algorithme, qui repose sur une approximation quadratique de la fonction objectif, s'accommode simplement et efficacement de contraintes non-linéaires ce qui représente un avantage décisif puisque la plupart des modèles sont soumis à de telles contraintes.

Cette méthode de calibrage constitue une contribution à part entière. Son utilisa-

tion permet de calibrer rapidement jusqu'à environ 8 paramètres d'un modèle de type file d'attente. Bien que cette méthode n'exige aucune forme particulière de résolution pour le modèle à calibrer, les méthodes de résolution analytiques sont préférées car elles permettent généralement une exécution plus rapide de l'algorithme DFO de recherche. Notons que cette méthode de calibrage est également une pièce essentielle de la méthode HLM puisque c'est elle qui est utilisée pour rechercher le calibrage des modèles génériques qui permet de reproduire au mieux les mesures du système étudié.

Les effets des propriétés distributionnelles d'ordre supérieur pour des files classiques

Enfin, le troisième axe des contributions de cette thèse traite de l'influence des propriétés distributionnelles d'ordre supérieur pour des files classiques de la théorie des files d'attente. Les résultats obtenus concernent notamment la file M/G/1 et les files multiserveurs.

Bien qu'il soit connu que le nombre moyen de clients dans une file M/G/1 ne dépende que des deux premiers moments de la distribution du temps de service (formule de Pollaczek-Khintchine), nous étudions si cette caractérisation du temps de service peut être suffisante afin de déterminer sans ambiguïté le nombre moyen de clients dans une file M/G/1/K ou dans une file multiserveur, comme semble le suggérer la grande majorité des approximations existantes. Plus généralement, nos travaux visent à mettre en évidence l'influence des propriétés d'ordre supérieur des distribution du temps de service et du temps entre les arrivées sur les paramètres de performances stationnaires (et vus à l'arrivée) pour des files classiques de la théorie des files d'attente.

Plan de la thèse

Ce rapport de thèse se compose de cinq chapitres et a été pensé de telle manière que certains chapitres puissent être lus indépendamment les uns des autres. Le Chapitre 1 présente les notions essentielles communes à la compréhension des chapitres suivants (paramètres de performances, charge de travail, jeu de mesures, etc).

Le Chapitre 2 décrit la méthode de calibrage automatique. Il comprend en plus de la présentation complète de son fonctionnement, d'une évaluation de ses performances, de deux cas d'étude facilitant sa compréhension, un état de l'art des solutions permettant de traiter la recherche du calibrage. Cet état de l'art nous permet de mettre en évidence la nécessité de développer un nouvel algorithme de recherche, qui est présenté dans ce chapitre et qui est entièrement pensé pour le calibrage d'un modèle de type file d'attente.

Le Chapitre 3 est consacré à la présentation de la méthode HLM. Le chapitre s'ouvre sur un état de l'art de méthodes existantes s'apparentant par certains aspects à la méthode de modélisation HLM. Le reste du chapitre décrit le fonctionnement de la méthode HLM.

Le Chapitre 4 contient des exemples d'applications de la méthode HLM sur des jeux de mesures provenant de systèmes informatiques et de communication réels et variés. Un effort particulier a été fait afin d'illustrer les utilisations possibles des modèles produits par la méthode et également afin de valider expérimentalement la capacité prédictive des modèles obtenus.

Enfin, le Chapitre 5 s'intéresse à l'influence des propriétés distributionnelles d'ordre supérieur pour des files classiques de la théorie des files d'attente comme par exemple les files M/G/1 et M/G/C.

Chapitre 1

Tour d'horizon des notions clés

Sommaire

Avant-propos

Ce chapitre constitue un préambule aux trois chapitres qui suivent. Dans un premier temps, nous y présentons quelques notions essentielles relatives aux systèmes comme aux modèles. En informatique, comme dans probablement tous les domaines, les mots peuvent recouvrir plusieurs sens. C'est pourquoi, avant d'entrer dans les détails des travaux de cette thèse, nous indiquons la signification exacte qu'auront dans ce rapport certains termes au sens parfois multiple. Nous introduisons également un vocabulaire plus spécifique, relatif aux travaux présentés dans ce rapport. Enfin, nous illustrons concrètement les objectifs de cette thèse avec deux exemples issus de situations réelles.

1.1 Introduction

La plupart des systèmes ou des réseaux informatiques peuvent être décrits comme des assemblages de ressources matérielles ou logicielles auxquelles tentent d'accéder un ensemble de clients ou de demandes. Comme nous l'avons dit précédemment, pour un ordinateur personnel ou un serveur central, les ressources matérielles peuvent représenter la mémoire principale, un processeur, les disques ou les terminaux. Les ressources logicielles peuvent correspondre aux accès bloquant en écriture à un fichier. Quant aux clients, ils peuvent représenter des processus, des requêtes disques ou des utilisateurs. Dans les réseaux informatiques, les ressources matérielles peuvent faire référence à un serveur Web, à des commutateurs, à des buffers ou à des liens physiques tandis qu'une ressource logicielle peut correspondre, par exemple, aux contraintes imposées par une fenêtre de contrôle d'émission sur un flot (comme le fait TCP). Selon les systèmes, les clients peuvent être des requêtes HTTP, des paquets IP, des trames ou des flots de données.

Les ressources d'un système sont décrites dans les *spécifications* du système. En les consultant, l'analyste s'informe sur les composants et le fonctionnement interne du système. En revanche, lorsqu'il s'agit de décrire le comportement d'un système, ce sont les paramètres de performances du système et ses conditions de charge, qui doivent être considérés.

1.2 Paramètres de performances

Les *paramètres de performances* permettent de décrire le comportement quantitatif d'un système ou d'un modèle. Leur nature dépend des systèmes considérés. Pour un serveur central, le taux d'utilisation de son(ses) processeur(s) peut être un paramètre

de performance important. Il définit la proportion de temps pendant laquelle ses ressources sont occupées. Dans les réseaux informatiques, un paramètre de performances important est le temps de réponse (i.e. délai d'acheminement) de bout-en-bout. Il mesure le temps qui sépare l'émission d'un message, de la réception par le destinataire. Pour les disques durs, le débit (également appelé le taux de transfert) constitue généralement un paramètre de performances important qui indique la quantité d'informations transmise par le disque par unité de temps (par exemple, en Mo/sec).

Parmi les paramètres de performances les plus fréquents, on trouve :

- le débit (de sortie) X, qui définit la vitesse à laquelle les clients quittent le système ;
- le temps de séjour R, qui définit le temps écoulé entre l'instant d'arrivée d'un client et la fin de son service ;
- le nombre de clients Q, qui définit le nombre de clients présents dans le système ;
- l'utilisation d'une ressource U, qui définit la proportion de temps pendant laquelle la ressource est occupée ;
- la probabilité de rejet Pr, qui définit le taux de clients refusés par le système.

Souvent, seule la valeur moyenne des paramètres de performances est considérée, $\bar{X}$, $\bar{R}$, $\bar{Q}$, et $\bar{U}$. Pourtant, il peut parfois être utile de connaître d'autres moments des paramètres de performances, ou bien même leur distribution complète. Par exemple, le nombre moyen de clients dans un système $\bar{Q}$ peut permettre de savoir qu'un système est en moyenne modérément chargé, mais ne permet pas de savoir si la capacité d'accueil maximale du système est souvent atteinte. C'est en considérant la variance, qui renseigne sur la dispersion de la variable par rapport à sa moyenne, ou plus généralement les moments d'ordre supérieur de Q, qu'on pourra y répondre.

Dans la suite de ce rapport, les paramètres de performance mesurés d'un système sont marquées par l'exposé mes pour les distinguer de ceux évalués sur un modèle, désignés eux, par l'exposé th. Ainsi, $\bar{R}^{mes}$ représente un temps de séjour mesuré sur un système et $\bar{R}^{th}$ représente un temps de séjour calculé par un modèle.

1.3 Charge de travail

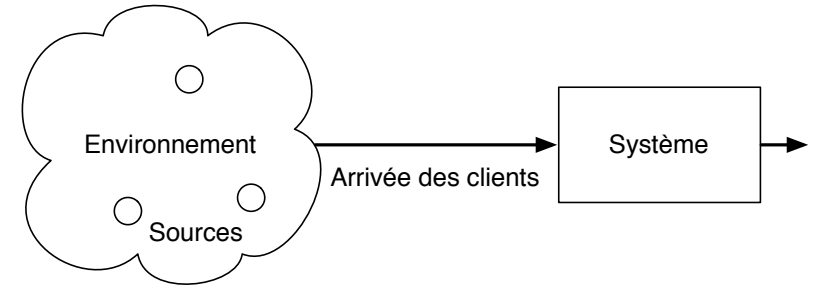
La *charge de travail* (ou charge) d'un système symbolise toutes les demandes de traitement destinées au système (ou modèle) considéré. Chaque demande de traitement est représentée par un client, qui rivalise potentiellement avec d'autres clients pour accéder aux ressources du système. Selon les systèmes, les clients peuvent représenter des processus, des requêtes disque, des taches, des requêtes HTTP ou encore des paquets

IP. Les clients sont émis par des entités que l'on nomme les *sources*. Les sources peuvent, par exemple, correspondre à des utilisateurs ou à des applications. L'ensemble des sources qui engendrent la charge de travail constitue l'*environnement* du système [28]. Le nombre de sources qui composent l'environnement varie selon les systèmes. Pour certains systèmes, les clients sont émis à partir d'une seule source. Par exemple, les taches destinées à un ordinateur personnel sont généralement engendrées par un seul utilisateur qui peut donc être considéré comme l'unique source du système. A l'inverse, sur d'autres systèmes, les clients peuvent être engendrés par plusieurs sources. Dans le cas d'un serveur central, plusieurs utilisateurs peuvent être connectés et engendrer simultanément des clients.

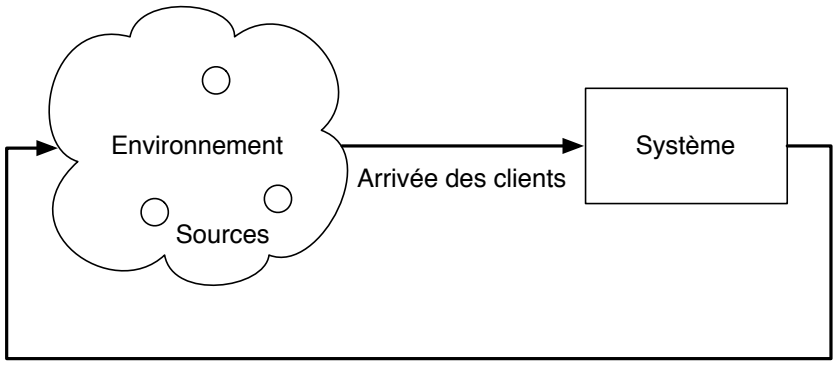
L'environnement d'un système peut être dépendant ou indépendant de l'état du système. Si l'activité des sources est indépendante de l'état du système, le système est décrit comme *ouvert*. Dans les systèmes ouverts, les clients émis par les sources sont traités par le système puis le quittent définitivement (cf. Figure 1.1(a)) [5]. Dans le cas contraire, lorsque l'état du système (rétro-)influence l'activité de l'environnement, le système est dit *fermé* : l'environnement et le système sont asservis. C'est par exemple le cas lorsque les clients émis par les sources sont traités par le système puis retournent dans l'environnement (cf. Figure 1.1(b)). Dans ces cas-là, le nombre de clients répartis entre le système et l'environnement est généralement limité et constant.

Les performances d'un système sont étroitement liées à la charge placée en entrée du système [47, 43, 36]. Elles s'établissent en réponse au niveau de la charge placée en entrée du système (ou à une hypothèse de charge dans le cas d'un modèle). Par exemple, le temps de séjour d'une requête HTTP dans un serveur Web dépend du nombre de requêtes présentes dans le système avant elle (en service ou en attente de service). Ainsi le temps de séjour moyen d'un client dans le serveur sera certainement plus court lorsque la charge soumise au serveur est faible.

Pour de nombreux systèmes et réseaux informatiques, la charge que produit leur environnement varie selon les secondes, les minutes, les heures, ou encore les journées considérées. Ces variations peuvent être dues à une augmentation (ou une diminution) du nombre de sources. Le niveau de la charge peut également changer lorsque l'intensité des sources augmente (ou diminue) ou lorsque la taille des clients (exprimée par exemple en besoin de temps de service) émis par les sources change. Les paramètres de performances du système réagissent à ces variations de la charge. Ainsi, l'observation (prolongée) d'un système peut conduire à plusieurs ensembles de mesures, correspondant chacun à des niveaux de charge différents. Par exemple, les arrivées de requêtes sur un serveur Web seront probablement plus nombreuses la journée que la nuit et par



(a) Modèle ouvert



(b) Modèle fermé

FIG. 1.1 – L'environnement et le système

conséquent, le temps de séjour d'une requête sera probablement plus long en journée. Si le serveur est instrumenté, les mesures permettront de connaître les performances du serveur sous différents niveaux de charge, qui eux peuvent être ou non mesurés.

1.4 Jeu de mesures

Les mesures considérées dans notre travail correspondent à des mesures du système étudié pris dans son ensemble. Elles concernent le comportement global du système et pas seulement une de ses parties[1]. Autrement dit, le système étudié est vu comme une boîte noire, pour laquelle on dispose uniquement de mesures d'entrée/sortie. Par exemple, les mesures d'entrée/sortie peuvent être le débit moyen en sortie du système $\bar{X}$, le temps de séjour moyen dans le système $\bar{R}$, le nombre de clients moyen présents dans tout le système $\bar{Q}$ ou encore la probabilité de rejet en entrée du système Pr (cf.

[1]Notons que cette hypothèse n'empêche naturellement pas le système d'être lui-même un sous-système d'un autre système.

Figure 1.2). A l'inverse, si le système étudié est un serveur central, le débit à la sortie de son processeur ne constitue pas une mesure d'entrée/sortie du système.

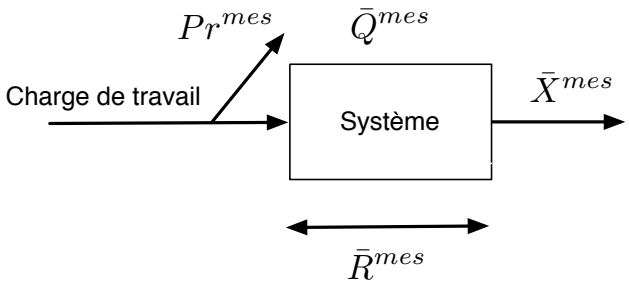


FIG. 1.2 – Les mesures d'entrée/sortie sur un système de type boîte noire

Notons qu'étant donné qu'on s'intéresse à des mesures d'entrée/sortie d'un système pris dans sa globalité, la loi de Little [65, 56, 2, 5] (i.e. $\bar{Q} = \bar{R}\bar{X}$) peut être appliquée. Ainsi, si l'on dispose de mesures pour $\bar{X}$ et $\bar{Q}$, on peut calculer directement $\bar{R}$.

Un *point de mesure* correspond au résultat d'une mesure. Il indique la valeur des paramètres de performance qui ont été mesurés. Même si un point de mesure est toujours associé à un niveau de charge (celui qui était à l'oeuvre lors de la mesure), celui-ci n'est pas toujours connu. Les points de mesure sont généralement représentés sous la forme de n-uplets. Par exemple, si le système considéré est un lien Ethernet qui achemine des paquets UDP entre deux postes, les mesures peuvent concerner le débit d'arrivée moyen des paquets au destinataire, $\bar{X}$ (qui peut être différent du taux d'émission des paquets s'il y a des pertes au cours de la transmission) ainsi que le délai d'acheminement moyen d'un paquet, $\bar{R}$. Dans ce cas, un point de mesure correspond à un couple de valeurs $(\bar{X}, \bar{R})$. Le point de mesure serait complet si en plus de $\bar{X}$ et de $\bar{R}$, le niveau de la charge lors de la mesure était renseigné. Ici, le niveau de charge correspond typiquement au taux d'émission des paquets UDP par le poste émetteur.

Un *jeu de mesures* est un ensemble de points de mesure. La mesure répétée d'un système occasionne généralement plusieurs points de mesure qui correspondent à des niveaux de charge différents. On parle parfois de *points de fonctionnement* du système. Par exemple, un jeu de mesures peut indiquer les paramètres de performances $\bar{X}$ et $\bar{R}$ lorsque le niveau de la charge en entrée du système étudié est faible, modéré puis fort.

1.5 Exemples de jeux de mesures

Nous présentons deux exemples de jeux de mesures qui ont été engendrés par des systèmes réels. Le premier exemple, extrait d'un article de Cao et al. [23], concerne les

performances d'un serveur Web Apache. Son débit, $\bar{X}^{mes}$, correspond au taux auquel les requêtes HTTP sont servies par le serveur. Le temps de séjour, $\bar{R}^{mes}$, désigne le temps moyen que passe une requête dans le serveur Apache. Le jeu de mesures, qui est présenté dans la Table 1.1 et illustré par la Figure 1.3, comporte cinq points de mesure de type : $(\bar{X}^{mes}, \bar{R}^{mes})$. Les mesures décrivent le comportement du serveur sous cinq conditions de charge différentes mais qui ne sont elles-mêmes pas connues. Il apparaît également sur cette figure que plus $\bar{X}^{mes}$ est important, plus $\bar{R}^{mes}$ est grand. Ce comportement, que l'on retrouve sur de nombreux autres systèmes et réseaux informatiques, traduit généralement la *congestion* progressive du système (et peut aller jusqu'à sa *saturation*).

$\bar{X}^{mes}$ (req/sec)	$\bar{R}^{mes}$ (sec)
80.0	1.89E-02
100.0	3.77E-02
120.0	5.66E-02
140.0	2.64E-01
140.4	1.43E00

TAB. 1.1 – Jeu de mesures pour un serveur Web Apache

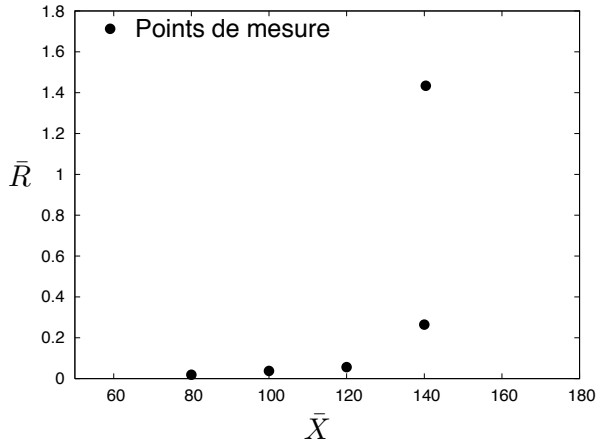


FIG. 1.3 – Jeu de mesures pour un serveur Web Apache

Le second exemple de jeu de mesures a été obtenu sur un serveur central. Plusieurs utilisateurs, représentant chacun une source de requêtes (i.e. clients), sont connectés au serveur et se partagent ses ressources. Les requêtes émises par les sources peuvent être immédiatement servies par le serveur ou contraintes à attendre si d'autres requêtes oc-

S^{mes}	$\bar{R}^{mes}$ (sec)
1	1.53E-03
2	1.67E-03
4	2.52E-03

TAB. 1.2 – Jeu de mesures pour un serveur central

cupent déjà les ressources du serveur. Le serveur central été mesuré pour trois nombres différents d'utilisateurs connectés, S^{mes}. A chaque fois, le temps de séjour moyen d'une requête dans le serveur, $\bar{R}^{mes}$ a été mesuré. Chaque point de mesure indique une valeur ($\bar{R}^{mes}$) et le S^{mes} associé. Ce jeu de mesures, présenté dans la Table 1.2, indique également qu'à mesure que le niveau de la charge augmente (ici c'est le nombre de sources qui augmente), le temps de séjour moyen des requêtes augmente.

1.6 Illustration des objectifs

Les notions définies ci-dessus permettent de préciser les objectifs de cette thèse. Comme nous l'avons dit, la méthode de modélisation automatique HLM, qui constitue le premier objectif de cette thèse, ne nécessite qu'un jeu de mesures pour fonctionner. Tout le processus de modélisation, c'est à dire le choix du modèle (et de son environnement) et son calibrage, doivent être décidés à partir de cette seule connaissance. Essentiellement, la méthode HLM recherche un modèle simple de type file d'attente dont le comportement reproduit les points de mesure du système. Plus précisément, en soumettant le modèle recherché à des niveaux de charge connus ou à inférer selon les jeux de mesures, ses paramètres de performances doivent être proches de ceux indiqués par les points de mesure. Ainsi, dans le cas du serveur Web Apache présenté ci-dessus, la méthode HLM recherche un modèle simple et correctement calibré tel que ses paramètres de performances $\bar{X}^{th}$ et $\bar{R}^{th}$ évalués pour certains niveaux de charge à inférer reproduisent ceux mesurés sur le système.

Le deuxième objectif de cette thèse concerne l'automatisation de l'étape de calibrage, qui constitue une partie importante de la méthode HLM. Il s'agit de développer une méthode générale et automatique pour déterminer la valeur des paramètres d'un modèle prédéfini (ou supposé) en fonction d'un jeu de mesures. La combinaison recherchée des paramètres doit permettre au modèle de reproduire au mieux, pour des

conditions de charge connues ou non, les performances données par chacun des points du jeu de mesures.

Chapitre 2

Le calibrage automatique d'un modèle

Sommaire

2.1 Introduction

Le calibrage constitue une étape essentielle à la modélisation. Dans le cas d'une approche constructive, elle succède généralement à la description du modèle. Le calibrage d'un modèle consiste à déterminer la valeur de ses paramètres.

La valeur de certains paramètres, les *paramètres déterminés*, que nous désignons par $(\alpha_1, \ldots, \alpha_n)$, peut être obtenue directement à travers la connaissance du système concerné. En général, cela suppose que deux conditions soient réunies. Premièrement, les paramètres doivent pouvoir être associés directement à certains aspects du système. Certains paramètres du modèle peuvent par exemple correspondre à des paramètres connus du système ou bien représenter certains de ses composants. Deuxièmement, une fois cette correspondance établie, les spécifications du système sont requises afin d'y rechercher les valeurs des paramètres.

A l'inverse, les *paramètres indéterminés*, désignés par $(\beta_1, \ldots, \beta_n)$, sont ceux dont la valeur est plus compliquée à obtenir. La raison de cette difficulté peut être liée à l'absence d'une des deux conditions requises pour les $(\alpha_1, \ldots, \alpha_n)$. Premièrement, il peut être difficile d'établir une correspondance entre un paramètre du modèle et ceux du système, par exemple, lorsque le paramètre du modèle agrège plusieurs composants du système. Deuxièmement, le manque de connaissances précises sur le système peut expliquer l'existence de paramètres indéterminés dans un modèle.

Pour déterminer les valeurs des paramètres indéterminés, $(\beta_1, \ldots, \beta_n)$, nous décidons de nous appuyer sur des mesures du système associé au modèle. Le principe consiste alors à choisir les $(\beta_1, \ldots, \beta_n)$ de façon à ce que le modèle ainsi calibré s'accorde au mieux à ces mesures. Il existe de nombreux travaux sur le calibrage d'un modèle présupposé à partir de mesures [69, 48, 53]. Les méthodes proposées diffèrent par leurs hypothèses sur la nature des mesures et sur le modèle à calibrer.

Dans ce chapitre, on suppose qu'on dispose d'une part, d'un jeu de mesures pour un système donné, et d'autre part, d'un modèle de type file d'attente non calibré. Rappelons qu'un jeu de mesures suppose que le système a été mesuré pour différents niveaux de charges (cf. Chapitre 1, page 19). Plus formellement, on suppose que l'on dispose de N points de mesure désignés par $(M_1, \ldots, M_N)$. Chaque M_i est un vecteur de p paramètres de performances qui correspondent tous à un niveau de charge donné du système.

Nous présentons une méthode automatique, simple à mettre en oeuvre et efficace pour décider des valeurs des paramètres indéterminés d'un modèle, $(\beta_1, \ldots, \beta_n)$, de façon à ce que le modèle ainsi calibré s'accorde au mieux aux N points de mesure.

L'exigence d'automatiser la méthode a certaines implications. Premièrement, la méthode ne peut faire appel à des propriétés liées spécifiquement au modèle considéré, à ses paramètres, ou à ceux du système. Deuxièmement, la méthode doit pouvoir fonctionner pour le plus grand nombre possible de types de jeux de mesures. Non seulement la nature et le nombre p des paramètres de performances des point de mesure M_i peuvent varier d'un exemple à un autre, mais le niveau de charge associé à chaque M_i doit pouvoir être supposé connu ou non. Par ailleurs, des contraintes sur les paramètres doivent pouvoir être ajoutées afin de rendre la méthode attractive à un plus grand nombre d'utilisateurs.

Notre choix a consisté à formuler le calibrage d'un modèle comme un problème d'optimisation avec contraintes (non-linéaires). Cette formulation suppose la définition d'une fonction objectif et d'un algorithme de recherche. Plusieurs définitions possibles de la fonction objectif sont examinées et comparées. Ensuite, un algorithme original de type DFO (« Derivatives Free Optimization ») permettant une gestion simple et efficace des contraintes est décrit, et ses performances sont évaluées et comparées à celles d'autres algorithmes. Enfin, nous présentons l'application de cette méthode sur deux exemples extraits de situations réelles.

2.2 Cas d'étude

Les deux cas d'étude présentés dans ce chapitre constituent des exemples-type pour notre méthode et aident à sa compréhension. Dans les deux cas, on dispose d'un jeu de mesures pour le système concerné et d'une proposition de modèle. Cependant, certains paramètres du modèle, $(\beta_1, \ldots, \beta_n)$, sont indéterminés.

2.2.1 Cas d'étude A

Le premier cas d'étude a été obtenu à partir de l'article de Cao et al. [23]. Il concerne les performances d'un serveur Web Apache. Le jeu de mesures, que nous considérons pour cet exemple, décrit le comportement du serveur pour cinq conditions de charge différentes. Chaque point de mesure associe un débit, $\bar{X}^{mes}$, qui correspond au taux auquel les requêtes HTTP sont servies par le serveur, à un temps de séjour, $\bar{R}^{mes}$, qui désigne le temps moyen que passe une requête dans le serveur Apache. En revanche, le niveau de la charge qui était à l'oeuvre lors de la mesure n'est pas renseignée. Le détail du jeu de mesures a été présenté précédemment dans le Chapitre 1 (cf. Table 1.1, page 21).

Cao et al. ont proposé une file M/G/1/K*PS [56] pour représenter le comportement

du serveur Web. Ce modèle doit leur permettre de reproduire, avec suffisamment de précision, la relation entre le débit des requêtes, $\bar{X}$, et leurs temps de séjour moyens dans le serveur, $\bar{R}$, telle qu'elle est reflétée par les mesures.

Dans une file M/G/1/K*PS, les arrivées des clients (ici les requêtes HTTP) suivent un processus de Poisson de taux λ. La discipline de service du serveur est de type « Processor Sharing » (i.e. Processeur Partagé) ce qui signifie que le serveur de la file sert tous les clients en même temps, mais avec une vitesse inversement proportionnelle au nombre de clients simultanément présents dans la file. La capacité d'accueil de la file K indique le nombre maximal de clients pouvant être présents au même instant dans la file. Lorsque la file est pleine, aucun nouveau client ne peut entrer dans la file : toute nouvelle arrivée est rejetée. Quant au temps de service du serveur, on suppose qu'il est distribué selon une loi générale. Toutefois, la connaissance de la moyenne du temps de service, τ, suffit dans ce cas car les autres moments du temps de service n'influent pas les performances moyennes de la file [56]. Pour une file M/G/1/K*PS, des expressions analytiques permettent d'obtenir de façon exacte le temps de séjour moyen, $\bar{R}^{th}$, et le débit moyen, $\bar{X}^{th}$ en régime permanent [56]. Avec $\rho = \lambda.\tau$, on a : $\bar{X}^{th} = \lambda\frac{1-\rho^K}{1-\rho^{K+1}}$ et $\bar{R}^{th} = \frac{\tau}{1-\rho}\frac{1-\rho^{K+1}}{1-\rho^K}$. Les valeurs de $\bar{X}^{th}$ et $\bar{R}^{th}$ dépendent de trois variables : le niveau de charge λ, la capacité d'accueil de la file K et le temps de service moyen du serveur τ.

Pour vérifier si la file M/G/1/K*PS permet de modéliser le comportement d'entrée/sortie du serveur, il faut décider de la valeur des paramètres structurels du modèle (τ et K) et ensuite comparer les performances de la file ainsi calibrée à celles du serveur Web.

Rapporté à notre contexte, l'objectif de ce cas d'étude consiste à décider de la valeur des deux paramètres indéterminés du modèle $(\beta_1, \beta_2) = (\tau, K)$ en s'appuyant sur un jeu de mesures obtenu sur le système concerné. Alors que les chercheurs de cet article ont estimé les valeurs de ces paramètres par une approche exhaustive, nous utilisons ce cas d'étude pour montrer, étape par étape, comment notre méthode peut décider efficacement du calibrage de ce modèle.

2.2.2 Cas d'étude B

Le second cas d'étude concerne un serveur central. Son fonctionnement et le jeu de mesures associé ont été présentés précédemment dans le Chapitre 1 (cf. Table 1.2, page 22). Rappelons que plusieurs sources, représentant chacune un utilisateur connecté au serveur, se partagent les ressources du serveur. Les requêtes émises par les sources peuvent être immédiatement servies par le serveur ou contraintes à attendre si d'autres requêtes occupent déjà les ressources du serveur. Le serveur central a été mesuré pour

trois nombres différents d'utilisateurs connectés, S^{mes}. A chaque fois, le temps de séjour moyen d'une requête dans le serveur, $\bar{R}^{mes}$ a été mesuré. Chaque point de mesure indique une valeur de $\bar{R}^{mes}$ et le S^{mes} associé.

La description du fonctionnement du serveur suggère le choix d'un modèle de type « machine repairman » (cf. Fig 2.1) pour le représenter. Les sources de ce modèle représentent les utilisateurs du système. Le temps séparant l'émission de deux requêtes par une source (« machine up time ») est distribué selon une loi exponentielle de moyenne γ. Ensuite, les requêtes engendrées par les sources entrent dans la file qui représente le serveur central. Il s'agit d'une file multiserveur avec C serveurs dont les temps de service (« machine repair time ») sont représentés par une loi exponentielle de moyenne t_s.

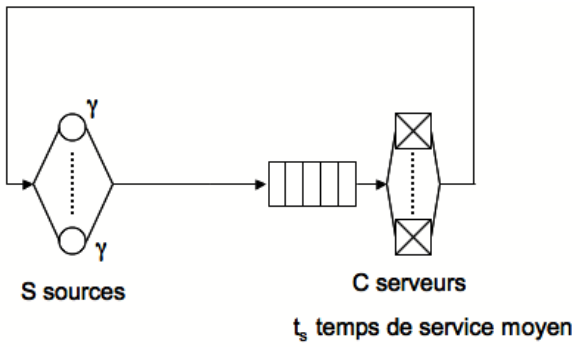


FIG. 2.1 – Le modèle de type « machine repairman »

Les paramètres de performances moyens de la file multiserveur peuvent être obtenus de façon exacte [56]. En régime permanent, le nombre de clients présents dans la file peut se modéliser par un processus de naissance et de mort. Nous avons représenté dans la Figure 2.2 la chaîne de Markov qui correspond à ce processus. Notons que μ désigne le taux de service de chacun des serveurs de file, soit $\mu = 1/t_s$ et S le nombre de sources du modèle.

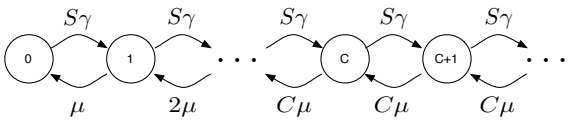


FIG. 2.2 – La chaîne de Markov associée au modèle « machine repairman »

Notre connaissance limitée du système ne permet pas de décider de la valeur des paramètres γ, C et t_s du modèle. C'est précisément l'objectif de notre méthode de calibrage que de décider de la valeur de ces trois paramètres indéterminés. Notons que

ce cas d'étude se distingue du précédent par la connaissance qu'on a des niveaux de charge ayant engendré les points de mesure. En réalité, cette connaissance sur la charge est partielle : les valeurs du paramètre de charge qui a changé d'un point de mesure à un autre (i.e. le nombre d'utilisateurs connectés au système) sont connues mais le temps moyen séparant l'émission des requêtes à chaque source γ reste à déterminer.

2.3 Les modèles intermédiaires

Les modèles de type file d'attente, à l'instar des deux exemples présentés ci-dessus, peuvent inclure à la fois des paramètres discrets et des paramètres continus. Cette dualité représente une difficulté car la grande majorité des travaux existants en optimisation concernent spécifiquement l'optimisation combinatoire ou l'optimisation continue. A notre connaissance, très peu d'algorithmes peuvent inclure à la fois des variables continues et des variables discrètes. Par conséquent, la première étape de notre méthode de calibrage consiste à rendre continus, du moins provisoirement, les paramètres discrets du modèle considéré.

Pour cela, nous avons défini le concept de « modèles intermédiaires » qui étend la définition des modèles d'origine en autorisant la présence de valeurs continues pour tous leurs paramètres. Les modèles intermédiaires doivent naturellement coïncider avec leur modèle d'origine lorsque les paramètres rendus continus ont une valeur discrète. La relaxation de la contrainte discrète (qui s'exerce sur les paramètres discrets d'un modèle) peut être faite de plusieurs façons.

Dans certains cas, la conversion peut être directe. Par exemple, pour le cas d'étude B, la méthode de résolution de la file reste « valable » si le nombre de serveurs C a une valeur réelle. Supposons que $C = 2.87$, alors il suffit de redéfinir certains taux de départ du processus de naissance et de mort associé à la file (cf. Figure 2.2) : le taux de départ de l'état 1 vers l'état 0 reste μ, celui de l'état 2 vers l'état 1 reste 2μ et en revanche, tous les taux de service des états $n > 2$ vers l'état $n - 1$ sont redéfinis à 2.87μ. Défini de la sorte, ce modèle intermédiaire de la « machine repairman » coïncide bien avec son modèle d'origine lorsque C est entier. Dans d'autres cas, la définition du modèle intermédiaire peut exiger d'apporter quelques remaniements à la résolution du modèle d'origine. Par exemple, pour le paramètre K de la file M/G/1/K*PS du cas d'étude A, nous avons ajouté un état supplémentaire à la chaîne de Markov associée au régime permanent du modèle. Le nouvel état $K+1$ est tel que le taux des arrivées depuis l'état K vers l'état $K+1$ est proportionnel à $K - \lfloor K \rfloor$, c'est-à-dire égal à 0.35λ si $K = 7.35$. Enfin, s'il apparaît compliqué de définir un modèle intermédiaire, et dans le soucis de

faire de cette méthode de calibrage une méthode automatique, le modèle intermédiaire peut être défini comme une simple interpolation linéaire du modèle d'origine entre ses valeurs discrètes.

Bien que dans sa forme la plus simple, la méthode que nous proposons recherche le calibrage du modèle intermédiaire, il est possible de la forcer à retourner un calibrage valable pour le modèle d'origine (avec des valeurs entières pour les paramètres discrets). Une façon très simple de le faire consiste à considérer tous les modèles pouvant être obtenus par combinaison des arrondis entiers (par excès et par défaut) des paramètres « artificiellement continus » et à garder comme calibrage pour le modèle d'origine, le meilleur d'entre eux (celui pour lequel la valeur de la fonction objectif, décrite ci-dessous, est la plus faible). Une autre façon, plus évoluée et que nous conseillons, consiste à relancer la méthode de calibrage sur le modèle intermédiaire après avoir converti, grâce au calibrage obtenu, un de ses paramètres « artificiellement continus » en constante entière (i.e. après avoir transformé un des β_i en un α_i). Notons que ces recherches « gigognes » représentent en général un effort supplémentaire limité puisque le nombre de paramètres du modèle est réduit et qu'on dispose, grâce au calibrage précédent, d'une idée assez précise de la solution optimale recherchée.

Dans la suite de cette section, le terme de modèle désignera le modèle intermédiaire si le modèle d'origine comporte des paramètres indéterminés à valeurs discrètes. Par conséquent, tous les paramètres indéterminés β_i correspondent à des paramètres continus.

2.4 Formulation du problème d'optimisation

Le calibrage d'un modèle consiste à décider de la valeur de ses paramètres indéterminés, $(\beta_1, \ldots, \beta_n)$. Dans notre cas, la recherche de ces valeurs repose sur l'existence d'un jeu de mesures obtenu sur le système associé au modèle.

Chaque calibrage possible d'un modèle constitue une *solution* de calibrage du modèle. Ainsi, une *solution* de calibrage correspond à un n-uplet $(\beta_1, \ldots, \beta_n)$ où chaque terme représente la valeur d'un paramètre indéterminé du modèle.

Par exemple, pour le cas d'étude A, les deux paramètres indéterminés, β_1 et β_2, sont le temps de service moyen du serveur τ et la taille du buffer K. Une solution de calibrage possible consiste à choisir (τ, K) égal à (6.25E-03, 250). Cette solution détermine un modèle calibré pour lequel on peut évaluer le temps de séjour moyen d'une requête, $\bar{R}^{th}$, et le débit atteint en sortie du modèle, $\bar{X}^{th}$. La Figure 2.3 montre les performances atteintes par ce modèle à tous les niveaux de charge λ possibles. Nous

avons également représenté sur cette figure les cinq points de mesures issus du serveur Web Apache correspondant à cinq niveaux de charge différents. Nous observons que, quels que soient les niveaux de charge en entrée du modèle, les performances moyennes du modèle associées à cette solution de calibrage, sont éloignées des deux dernières mesures prises sur le serveur Web Apache.

Afin de faciliter la lecture de cette section, le terme de *solution* fait ici implicitement référence à une solution de calibrage du modèle tandis que le terme de *modèle* désigne un modèle calibré.

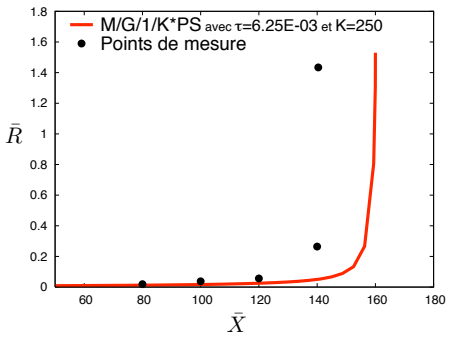


FIG. 2.3 – Un calibrage possible de la file M/G/1/K*PS

2.4.1 La fonction objectif

Dans un problème d'optimisation, la première étape consiste généralement à définir une fonction objectif. la *fonction objectif*, notée δ, doit permettre d'apprécier la qualité des solutions. C'est en définissant δ et ses contraintes, que l'on fixe les critères d'évaluation pour décider de la qualité d'une solution. La fonction objectif détermine donc, comme son nom l'indique, l'objectif poursuivi par la recherche. Ensuite, c'est à l'algorithme de recherche de découvrir la « meilleure » solution selon ces critères.

Dans notre contexte d'étude, la qualité d'une solution est évaluée vis-à-vis des mesures, puisque l'on suppose que ce sont elles qui permettent de calibrer un modèle. Plus précisément, la fonction objectif est définie autour de deux aspects essentiels. Le premier aspect, d'ordre qualitatif, concerne la *cohérence* de la solution avec les mesures. Cette exigence est assurée grâce à un ensemble de contraintes sur la fonction objectif (cf. Section 2.4.2). Le deuxième aspect est plus quantitatif. Il concerne la *proximité* de la solution vis-à-vis des mesures et a fortiori, la capacité du modèle concerné à reproduire les points de mesure $(M_1, \ldots, M_N)$. C'est le résultat de l'évaluation de δ qui détermine ce critère (cf. Section 2.4.3). La fonction objectif δ s'exprime donc sous la

forme suivante :

$$\delta = \begin{cases} \mathbb{R}^n & \rightarrow & \mathbb{R} \\ (\beta_1, \ldots, \beta_n) & \rightarrow & \delta(\beta_1, \ldots, \beta_n) \end{cases}$$

2.4.2 Les contraintes

La fonction objectif δ est sujette à plusieurs *contraintes*. Ces contraintes s'expliquent par le fait que certaines solutions de calibrage, $(\beta_1, \ldots, \beta_n)$, ne sont pas des candidats envisageables au calibrage du modèle. Nous distinguons trois types de contraintes dont les causes et les effets diffèrent.

Premièrement, certaines solutions constituent des calibrages « insensés » pour le modèle considéré : ces solutions incluent des valeurs irrationnelles pour les paramètres qu'elles représentent. Par exemple, si un paramètre β_i représente un temps de service moyen, sa valeur ne peut pas être négative. Généralement, les contraintes de ce type sont linéaires et elles définissent ensemble le domaine de définition de δ.

Le deuxième type de contraintes est lié aux points de mesure $(M_1, \ldots, M_N)$. Les contraintes de ce type visent à assurer la cohérence des solutions $(\beta_1, \ldots, \beta_n)$ avec les mesures[1]. Contrairement au premier type de contraintes, une solution qui viole une de ces contraintes correspond à un calibrage faisable mais incohérent avec les mesures. Autrement dit, la solution n'est pas en dehors de l'espace de définition de δ (i.e. $\delta((\beta_1, \ldots, \beta_n))$ peut être évalué) mais on la refuse comme solution finale pour le calibrage. Cette distinction est importante car elle est exploitée par notre algorithme de recherche (cf. Section 2.5).

En pratique, la plupart de ces contraintes s'obtiennent par des limites théoriques sur les paramètres de performances du modèle. Par exemple, pour le cas d'étude A, le temps de séjour moyen d'un client dans la file M/G/1/K*PS est nécessairement compris entre τ et $K\tau$ quel que soit le niveau de charge en entrée, λ. Par conséquent, une solution donnée de calibrage, (τ, K), est incohérente avec les mesures dès lors qu'un temps de séjour mesuré sur le serveur Web Apache se trouve à l'extérieur de l'intervalle $[\tau, K\tau]$. Des contraintes similaires peuvent être obtenues pour la plupart des paramètres de performances des modèles. Concernant le débit moyen en sortie de la file, une solution (τ, K) est incohérente avec les mesures si le seuil de saturation du modèle ainsi calibré ne se situe pas au-delà de tous les points de mesure $(M_1, \ldots, M_N)$. Nous présentons toutes les contraintes de ce type pour les deux cas d'étude A et B dans la Section 2.6 de ce chapitre.

[1] Si aucune solution ne permet de satisfaire ces contraintes, c'est probablement que le modèle lui-même n'est pas approprié.

Deux remarques relatives aux contraintes liées aux mesures méritent d'être soulignées. Premièrement, la plupart de ces contraintes n'étant pas linéaires, la formulation du problème de calibrage d'un modèle débouchera sur un problème d'optimisation non-linéaire. Deuxièmement, les incertitudes possibles et les biais intrinsèques à la mesure sont prises en compte en relâchant ces contraintes d'un certain degré (les détails d'implantation sont présentés dans la Section 2.6).

Il peut également exister des contraintes d'un troisième type pour δ. Elles sont généralement le résultat d'une connaissance partielle du système. Bien qu'étant des paramètres indéterminés, on peut forcer la valeur de certains paramètres, β_i, à être comprise dans un intervalle donné. En autorisant les contraintes de ce type, on élargit les fonctionnalités de notre méthode de calibrage.

Les contraintes qui s'exercent sur δ sont de nature variées. Certaines sont linéaires, et d'autres non-linéaires. Certaines contraintes fixent le domaine de définition de δ tandis que d'autres visent à restreindre l'espace de recherche des solutions afin d'assurer la cohérence des solutions avec les mesures (ou d'intégrer une connaissance supplémentaire sur le système). Les spécificités de ces contraintes impactent directement la recherche de la solution de calibrage d'un modèle et, comme nous le verrons dans la Section 2.5, elle nous ont conduit à développer un algorithme d'optimisation « sur mesure ».

2.4.3 La définition de δ

Dans cette section, nous décrivons, étape par étape, un procédé pour obtenir une fonction objectif, δ, adaptée à nos besoins. Les définitions obtenues sont valables pour des jeux de mesures de nature variée, ce qui constitue un atout pour l'automatisation de la méthode de calibrage.

Comme nous l'avons vu, l'évaluation de la fonction objectif doit permettre d'apprécier la capacité d'une solution (i.e. un modèle calibré par cette solution), à reproduire des points de mesures $(M_1, \ldots, M_N)$. On estime cette capacité en quantifiant la distance qui sépare les performances délivrées par le modèle de celles exprimées par le jeu de mesures.

Une des difficultés majeures liée à la définition de δ tient au fait que les mesures correspondent à des points, tandis que les performances d'un modèle peuvent être matérialisées par des ensembles de points ou par des courbes, selon que la charge du modèle est discrète ou continue. Lorsque la charge est continue, comme pour le cas d'étude A (cf. Figure 2.3)), le calcul de δ exigerait alors d'estimer la distance d une courbe à des points.

Du fait de cette difficulté, nous décidons de considérer uniquement N points de

fonctionnement possibles du modèle. Chacun de ces points correspond à un niveau de charge donné pour le modèle et s'associe à un point de mesure. C'est pourquoi, ces N points, que nous notons $(C_1, \ldots, C_N)$, sont nommés les *points couplés* du modèle. Grâce à ces points couplés, on peut définir simplement δ : évaluer la distance entre les points de mesure $(M_1, \ldots, M_N)$ et les points couplés $(C_1, \ldots, C_N)$.

Le choix de δ se déroule en deux étapes. La première étape consiste à sélectionner les points couplés C_i. La seconde étape est de définir une métrique qui quantifie la distance séparant les points couplés des points de mesure. Ces deux étapes, bien qu'étant indépendantes, sont toutes deux déterminantes pour le calcul de δ.

Afin de pouvoir être le plus général possible, nous définissons quelques notations supplémentaires. Les points couplés, $(C_1, \ldots, C_N)$, sont au nombre de N puisque, par définition, on associe un point couplé à chaque point de mesure. A l'instar des points de mesures $(M_1, \ldots, M_N)$, chaque point couplé C_i est un vecteur de p paramètres de performances. Ainsi, on a $M_i = (P_{i,1}^{mes}, \ldots, P_{i,p}^{mes})$ et $C_i = (P_{i,1}^{th}, \ldots, P_{i,p}^{th})$, où $P_{i,k}^{mes}$ et $P_{i,k}^{th}$ désignent respectivement le k^{eme} paramètre de performances d'un point de mesure et d'un point couplé du modèle. Le niveau de charge associé à chaque point de mesure M_i (resp. point couplé C_i) est noté L_i^{mes} (resp. L_i^{th}).

Les points couplés. La sélection des points couplés $(C_1, \ldots, C_N)$ s'effectue différemment selon que les niveaux de charge des mesures $(M_1, \ldots, M_N)$ sont connues ou pas.

1. **La charge est connue.**
   Lorsque les niveaux de charge des points de mesure sont connus, les points couplés sont dérivés directement des performances du modèle lorsqu'il est soumis à ces mêmes niveaux de charge. Plus formellement, on choisit les C_i tels que $L_i^{th} = L_i^{mes}$. Dans le cas d'étude B, on connaît les niveaux de charge, i.e. le nombre de sources connectées au serveur, qui étaient à l'oeuvre pour chacun des temps de séjour moyens mesurés. Les points couplés sont directement obtenus en évaluant le temps de séjour moyen d'un client dans le modèle pour ces mêmes nombres de sources. Ainsi, on a pour les points couplés, $C_i = (\bar{R}_i^{th})$ avec $L_i^{th} = L_i^{mes} = S_i^{mes}$.
2. **La charge est inconnue.**
   A présent on considère le cas où les niveaux de charge des points de mesure ne sont pas connus. Par conséquent, les points couplés ne peuvent pas être dérivés des performances du modèle lorsqu'il est soumis aux mêmes niveaux de charge.

   * *Le point le plus proche.*
     Une première façon de faire consiste à choisir les points couplés comme étant

les états du modèles dont les performances sont les plus « proches » des mesures. La notion de proximité a ici un sens arithmétique. On la quantifie avec une norme, par exemple Euclidienne. Ainsi, les C_i sont choisis de façon à rendre $||M_i, C_i||$ aussi petit que possible.
Cette sélection, combinée à la métrique Euclidienne qu'on présente dans la section suivante (cf. Formule (2.1)), correspond généralement à la méthode dite des moindres carrés. Bien qu'elle soit souvent utilisée [12, 10], y compris pour calibrer des modèles de type file d'attente [23], cette méthode présente quelques inconvénients. Premièrement, le choix des points couplés dépend clairement de l'unité dans laquelle sont exprimés les paramètres de performances. Deuxièmement, la recherche des C_i peut s'avérer difficile du fait qu'il n'existe, en général, aucune relation simple entre le niveau de charge d'un C_i, et sa proximité au M_i associé. La Figure 2.4 illustre le fonctionnement de cette approche pour le cas d'étude A, où $M_i = (\bar{X}_i^{mes}, \bar{R}_i^{mes})$.

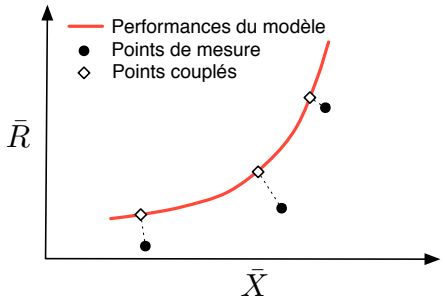


FIG. 2.4 – Les points couplés sont les points les plus proches des mesures

* *Un paramètre de performance en commun : le paramètre-critère.*
Une autre façon de choisir les points couplés consiste à s'appuyer sur un des paramètres de performances mesurés que l'on désigne comme le *paramètre-critère*. Les points couplés sont sélectionnés de façon à ce que leur valeur pour le paramètre-critère égale celle des points de mesure. Plus formellement, si P_k^{mes} représente le paramètre-critère, les C_i sont sélectionnés tels que $P_{i,k}^{th} = P_{i,k}^{mes}$. Si plusieurs points du modèle sont possibles, on pourra faire appel à un deuxième paramètre-critère.
L'inférence du niveau de charge L_i^{th}, afin de découvrir un état du modèle qui satisfait l'égalité $P_{i,k}^{th} = P_{i,k}^{mes}$, est souvent plus simple à effectuer que dans l'approche précédente où l'on recherche les performances les plus « proches » du modèle. Cela tient au fait qu'un seul paramètre de performance est

concerné, et que pour un certain nombre de systèmes à attente, les paramètres de performances ont une croissance ou une décroissance monotone avec le niveau de la charge. Une technique comme la bisection suffit alors à découvrir efficacement le niveau de charge du C_i recherché. Notons que l'existence des C_i est garantie par l'existence des contraintes décrites dans la Section 2.4.2. On illustre sur la Figure 2.5 le fonctionnement de cette approche pour le cas d'étude A en supposant que $\bar{X}$ est choisi comme paramètre-critère. Les C_i sont alors des couples $(\bar{X}_i^{th}, \bar{R}_i^{th})$ tels que $\bar{X}_i^{th} = \bar{X}_i^{mes}$.

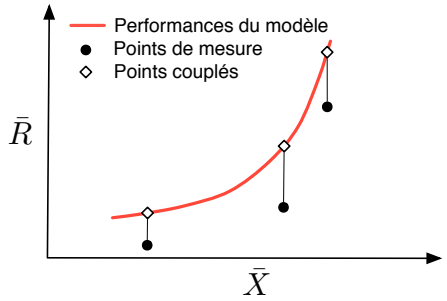


FIG. 2.5 – Les points couplés ont un paramètre en commun avec les mesures

Une métrique pour la distance. Une fois les points couplés choisis, on fait appel à une *métrique* pour quantifier la distance entre les performances du modèle (reflétées par les points couplés) et celles des points de mesure.

1. **Une métrique Euclidienne.**
   La façon la plus naturelle est peut-être de mesurer cette distance en sommant les distances entre chaque paire (M_i, C_i). Ces distances peuvent être calculées à l'aide d'une norme mathématique, comme par exemple la distance Euclidienne.

$$\delta = \sum_{i=1}^{N} ||M_i, C_i|| = \sum_{i=1}^{N} \sqrt{\sum_{k=1}^{p} (P_{i,k}^{mes} - P_{i,k}^{th})^2} \tag{2.1}$$

   Ce calcul de δ a l'inconvénient majeur de sommer ensemble des termes non homogènes. A cause de cette hétérogénéité dans les unités, il est difficile de garantir que les écarts entre M_i et C_i selon chaque paramètre de performances soient d'égale importance lors de l'évaluation de δ. Ainsi, selon les unités des paramètres de performances, les écarts sur certains paramètres de performances peuvent être exagérément pris en compte, tandis que d'autres peuvent être presque sans effet. Ces inégalités peuvent conduire à des appréciations « injustes » de certaines

solutions de calibrage. Pour les mêmes raisons, l'utilisation de cette métrique ne permet pas de privilégier volontairement les écarts sur certains paramètres de performances de façon à favoriser la proximité du calibrage recherché à un paramètre de performances en particulier.

Pour le cas d'étude A, la définition Euclidienne de la métrique donne la définition suivante pour δ :

$$\delta = \sum_{i=1}^{N} \sqrt{(\bar{X}_i^{mes} - \bar{X}_i^{th})^2 + (\bar{R}_i^{mes} - \bar{R}_i^{th})^2} \tag{2.2}$$

2. **Une métrique multi-critères.**
   Une meilleure solution consiste à sommer séparément toutes les distances observées sur chacun des paramètre de performances pour les paires (M_i, C_i). De la sorte, on évite certains inconvénients liés à la métrique Euclidienne. On introduit un poids γ_k pour chacun des paramètres de performances, ce qui permet de pondérer individuellement les effets de leurs écarts sur le résultat de δ. En faisant varier la valeur de ces coefficients pondérateurs, on peut facilement accroître (ou décroître) l'importance des écarts observés sur certains paramètres de performances dans l'appréciation que fait δ des calibrages. Avec cette métrique multi-critères, δ se définit plus formellement comme :

$$\delta = \sum_{k=1}^{p} \gamma_k \sum_{i=1}^{N} |P_{i,k}^{mes} - P_{i,k}^{th}| \tag{2.3}$$

Pour le cas d'étude A, en considérant que les deux paramètres de performances, $\bar{X}$ et $\bar{R}$, sont d'égale importance pour le choix du calibrage, on fixe alors $\gamma_1 = \gamma_2 = 1$ et on obtient pour δ :

$$\delta = \sum_{i=1}^{N} \left(|\bar{X}_i^{mes} - \bar{X}_i^{th}| + |\bar{R}_i^{mes} - \bar{R}_i^{th}| \right) \tag{2.4}$$

Au passage, notons que lorsque les points couplés sont sélectionnés de façon à avoir un paramètre de performances en commun avec les points de mesures, alors un terme de la somme dans la définition (2.3) de δ devient nul. Par exemple, si $\bar{X}$ est choisi comme paramètre-critère dans le cas d'étude A, la définition de δ

devient :

$$\delta = \sum_{i=1}^{N} |\bar{R}_i^{mes} - \bar{R}_i^{th}| \tag{2.5}$$

3. **Perfectionnements des métriques.**
   Jusqu'ici les définitions présentées pour la fonction objectif δ ((2.1), (2.3) et (2.4)) reposent uniquement sur des déviations absolues entre les points couplés et les points de mesure. Cette façon de faire peut être inadaptée lorsque les paramètres de performances ont des valeurs très différentes d'un point de mesure à un autre. C'est typiquement le cas des systèmes à attente. Par exemple, pour le cas d'étude A, la valeur de $\bar{R}_i^{mes}$ s'échelonne entre 19 ms et 1430 ms selon les points de mesure. Naturellement, un écart de 10 ms pour le $\bar{R}_i^{th}$ d'un point couplé n'a pas la même importance selon que la valeur du $\bar{R}_i^{mes}$ associé est de l'ordre de la dizaine ou du millier de ms. Plus généralement, en sommant uniquement les distances absolues entre les C_i et les M_i (quelle que soit la métrique utilisée), des écarts modérés pour des valeurs élevées de $P_{i,k}^{mes}$ et $P_{i,k}^{th}$ peuvent avoir un impact excessif sur le calcul de δ. Et inversement, si on somme uniquement des distances relatives, de faibles écarts entre $P_{i,k}^{mes}$ et $P_{i,k}^{th}$ peuvent peser trop fort dans le calcul de δ si leurs valeurs sont très petites. C'est pourquoi, on préconise de tenir compte à la fois des distances relatives et absolues dans le calcul de δ. Un paramètre, θ, dont la valeur est fixé entre 0 et 1 permet de décider de l'importance de chaque contribution. En intégrant ce changement à la métrique multi-critères, δ devient :

$$\delta = \sum_{k=1}^{p} \gamma_k \sum_{i=1}^{N} \left(\theta \frac{|P_{i,k}^{mes} - P_{i,k}^{th}|}{\hat{P}_k^{mes}} + (1-\theta) \frac{|P_{i,k}^{mes} - P_{i,k}^{th}|}{P_{i,k}^{mes}} \right) \tag{2.6}$$

où $\hat{P}_k^{mes} = \sum_{i=1}^{N} P_{i,k}^{mes}/N$ correspond à la valeur moyenne du k^{eme} paramètre de performances mesuré.

Un autre perfectionnement pour δ vise à tenir compte des incertitudes inhérentes à la mesure. On associe à chaque point de mesure un coefficient pondérateur w_i dont la valeur permet de refléter le niveau de confiance accordé à la mesure. Si on a quelques réserves sur l'exactitude d'un point de mesure, il suffit de diminuer le w_i associé pour réduire son impact sur le calcul de δ. En prenant en compte

les coefficients w_i, δ devient :

$$\delta = \sum_{k=1}^{p} \gamma_k \sum_{i=1}^{N} w_i \left(\theta \frac{|P_{i,k}^{mes} - P_{i,k}^{th}|}{\hat{P}_k^{mes}} + (1-\theta) \frac{|P_{i,k}^{mes} - P_{i,k}^{th}|}{P_{i,k}^{mes}} \right) \tag{2.7}$$

Notons que ces coefficients pondérateurs w_i peuvent également être utiles si l'on souhaite privilégier la recherche du calibrage sur une zone de fonctionnement particulière du système. Pour cela, il suffit d'augmenter les w_i associés aux points de mesure de ce domaine.

Pour le cas d'étude A, si $\bar{X}$ est choisi comme paramètre-critère pour choisir les points couplés, la définition de δ est :

$$\delta = \sum_{i=1}^{N} w_i \left(\theta \frac{|\bar{R}_i^{mes} - \bar{R}_i^{th}|}{\hat{R}^{mes}} + (1-\theta) \frac{|\bar{R}_i^{mes} - \bar{R}_i^{th}|}{\bar{R}_i^{mes}} \right) \tag{2.8}$$

4. **Une métrique basée sur une aire.**

   Les métriques présentées jusqu'ici peuvent délivrer des résultats « injustes » lorsque le jeu de mesures contient des « clusters » encore appelés amas de mesures. Un *cluster* désigne un groupe de points de mesure qui sont situés proches les uns des autres. Les points de mesure qui appartiennent à un même cluster ont par définition des valeurs semblables pour leurs paramètres de performances.

   Lorsqu'on observe un système en exploitation pendant une période de temps donnée, les conditions de charge du système peuvent se répéter. On peut obtenir, selon les cas, un jeu de mesures dont un certain nombre de points de mesure correspondent à un même point de fonctionnement du système. Ce point de fonctionnement est alors sur-représenté dans le jeu de mesures et forme un cluster.

   Les points qui composent un cluster ont tendance à peser trop fort sur le résultat de la fonction objectif δ, et ainsi à privilégier la zone de fonctionnement du cluster dans la recherche de calibrage du modèle. En effet, pour toutes les métriques proposées ci-dessus, un cluster agit comme un « super-point » de mesure, dont le poids est environ proportionnel au nombre de points qui le compose. C'est pourquoi la présence d'un cluster ou de plusieurs clusters dans un jeu de mesures incite le calibrage recherché du modèle à coïncider aux mesures du cluster plutôt qu'à être proche des autres points de mesure. Or, mis à part qu'il correspond à un point de fonctionnement fréquent du système pendant la période d'observation, un cluster n'apporte pas beaucoup plus d'information qu'un point de mesure seul.

   Plusieurs solutions sont possibles pour contre-balancer l'effet d'un cluster sur δ.

Mais pour la plupart, ces solutions se concilient mal avec une automatisation du calibrage. On peut par exemple contenir les effets néfastes d'un cluster sur le calcul de δ en fusionnant les points de mesure qui composent un cluster, en utilisant à bon escient les coefficients pondérateurs w_i, ou bien même en supprimant simplement les points redondants d'un cluster. Toutefois, une autre solution doit être trouvée lorsqu'on souhaite calibrer automatiquement un modèle à la volée. Cela peut être le cas si le système considéré produit des mesures en ligne.

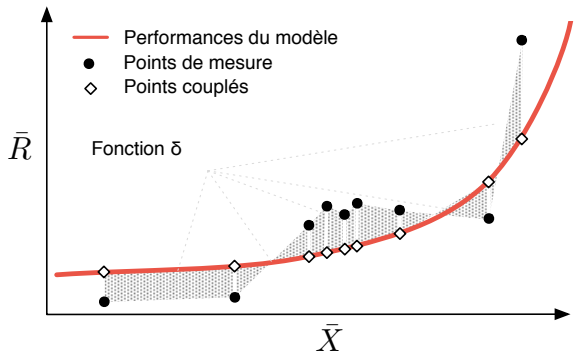


FIG. 2.6 – Une métrique basée sur une aire pour δ

Lorsqu'un jeu de mesures contient ou est susceptible de contenir un cluster, nous préconisons l'utilisation d'une métrique originale, qui repose sur une estimation de l'aire comprise entre les points de mesure et les paramètres de performances du modèle considéré. Notons que cette métrique, comme les précédentes, repose sur la définition de points couplés, C_i. Tous les éléments détaillés ci-dessus pour la sélection des C_i sont également valables ici. L'évaluation de δ avec cette métrique se déroule en deux étapes. Premièrement, on calcule les aires des quadrilatères formés par deux points de mesure consécutifs, i.e. M_i et M_{i+1}, et leur points couplés associés, i.e. C_i et C_{i+1}. Ensuite, le résultat de δ est obtenu en additionnant les aires de ces quadrilatères. La Figure 2.6 illustre le fonctionnement de cette métrique pour un jeu de mesures similaire à celui du cas d'étude A mais contenant cette fois-ci un cluster. En utilisant cette métrique pour évaluer la fonction objectif, les risques liés aux clusters disparaissent naturellement car l'importance d'un point de mesure croît avec son « degré d'isolement ».

2.5 Calcul des paramètres

Cette section est consacrée à l'algorithme de recherche qui constitue, avec la fonction objectif, un des éléments essentiels à notre méthode de calibrage automatique. Les sections précédentes de ce chapitre ont permis de formuler la recherche du calibrage d'un modèle comme un problème d'optimisation continue. A présent, et c'est l'objet de cette section, il reste à décider d'un algorithme qui permette de rechercher efficacement la solution qui minimise (la valeur de) la fonction objectif δ.

La Figure 2.7 représente un exemple possible de fonction objectif δ pour un modèle avec deux paramètres indéterminés (β_1, β_2). Les coordonnées du minimum de cette fonction correspondent au calibrage recherché du modèle.

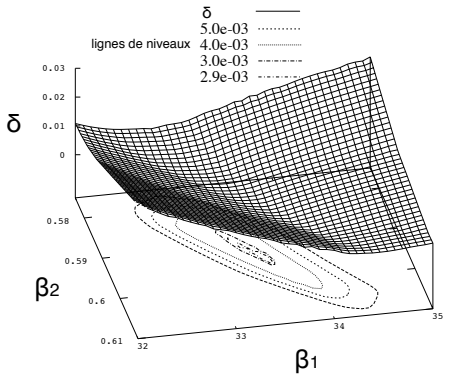


FIG. 2.7 – Un exemple de fonction δ pour deux paramètres indéterminés

2.5.1 Le choix d'un algorithme d'optimisation.

La plupart des méthodes d'optimisation continue dérivées de l'analyse fonctionnelle ou de l'analyse numérique ne sont pas adaptées à notre problème, c'est-à-dire aux spécificités liées à la fonction objectif δ et à ses contraintes. Dans notre cas, la fonction δ peut être une fonction non-linéaire, non-convexe, non-dérivable et non-continue.
L'*analyse fonctionnelle* permet la définition de méthodes exactes et efficaces pour trouver l'optimum d'une fonction. Ces méthodes s'appuient généralement sur une expression analytique des dérivées de la fonction. Certaines font également appel à la matrice hessienne de la fonction (associées aux dérivées secondes partielles) afin de déterminer la nature des points critiques de la fonction (i.e. les points d'annulation du gradient). Cependant, ce type de méthode est inadapté à notre problème de calibrage pour plusieurs raisons. Premièrement, si l'on souhaite que la méthode de calibrage puisse fonctionner

pour le plus grand nombre de modèles, aucune propriété sur la fonction objectif δ ne peut être supposée. En particulier, on ne peut pas supposer que la fonction objectif est continûment dérivable. Deuxièmement, ces méthodes nécessitent le calcul des expressions analytiques des dérivées de la fonction. En plus de constituer un préalable en contradiction avec le principe d'automatisation de la méthode de calibrage, l'obtention de ces expressions peut être compliquée, voire impossible si les paramètres de performances d'un modèle n'ont pas d'expression analytique connue.

Les méthodes d'*analyse numérique* se divisent en deux catégories. Les *méthodes directes*, qui constituent la première catégorie, permettent de résoudre de façon exacte et en temps polynômial certains problèmes de mathématique continue. Cependant, il n'existe généralement pas de méthodes directes possibles pour les problèmes non-linéaires [58]. Ainsi, ce type de méthodes est inadapté pour rechercher le minimum d'une fonction comme δ, qui est très certainement non-linéaire dans la majorité des cas.

Les *méthodes itératives* constituent généralement le seul choix possible pour résoudre les problèmes non-linéaires. Un nombre important d'entre elles s'appuient sur le calcul des dérivées locales pour se rapprocher par itérations successives de l'optimum. Les algorithmes de meilleure descente (i.e. descente de gradient), du gradient conjugué, de Newton [4] ou encore de Levenberg-Marquardt [64] sont quelques unes des plus connues. Ces algorithmes sont réputés très efficaces quand on a affaire à un problème d'optimisation convexe. On peut toutefois utiliser ces algorithmes lorsque, comme c'est le cas pour δ, la fonction objectif n'est pas convexe. Ceci étant dit, plusieurs raisons font obstacle à leur utilisation pour notre méthode de calibrage. Premièrement, ces algorithmes reposent sur l'existence des dérivées (premières ou secondes) de la fonction objectif pour pouvoir fonctionner correctement. Or, cette hypothèse ne peut être garantie si l'on souhaite rendre la méthode de calibrage la plus générale possible. Deuxièmement, les régions plates, fréquentes dans δ (cf. Section 2.6.6), constituent des passages difficiles pour ces algorithmes. Puisque le gradient est par définition nul ou quasi-nul dans ces régions, ces algorithmes ne trouvent pas la bonne direction de recherche [79, 78]. Troisièmement, en supposant que les dérivées de δ existent, leurs calculs approchés peut poser problème. D'une part, le calcul de δ implique un calcul, onéreux en temps et en ressources, car il contient une somme de N termes où chacun des termes requiert la résolution du modèle, voire plusieurs si l'inférence du niveau de charge des points couplés n'est pas directe. Ainsi, l'estimation locale des dérivées de δ peut conduire à des coûts calculatoires considérables. D'autre part, l'estimation des dérivées peut être biaisée par des arrondis numériques dans le calcul de la fonction δ. Par exemple, nous

avons expérimenté l'algorithme SolvOpt, qui est basé sur une implantation de l'algorithme de Schor's [50]. Cette routine estime les dérivées premières de δ pour guider sa recherche. Les résultats obtenus ont montré des taux de convergence assez faibles. Nous avons remarqué que les gradients calculés par SolvOpt sont souvent mal dirigés. Ces erreurs résultent, selon nous, de la présence de régions plates et d'approximations numériques sur le calcul δ qui compromettent le calcul des gradients.

Du fait des remarques précédentes, nous avons opté pour des méthodes numériques itératives qui ne font pas appel aux dérivées (ni de façon exacte, ni de façon approchée). On parle alors d'algorithmes DFO (« derivative-free optimization ») [83, 85]. Plus généralement, ces algorithmes ne font aucune hypothèse sur les propriétés de la fonction objectif et sont réputées efficaces lorsque le calcul de la fonction objectif est coûteux en temps et en ressources. Le plus connu de ces algorithmes est peut-être celui du « downhill simplex » (aussi connu sous le nom de la méthode Nelder-Mead) [76]. Nous l'avons expérimenté, mais les résultats ont révélé des problèmes de convergence. Une des raisons probables à ces problèmes est que le simplex cherche à évaluer δ en dehors de son domaine de définition, i.e. pour des $n-uplets$ $(\beta_1, \ldots, \beta_n)$ qui violent certaines contraintes.

Plusieurs autres routines d'optimisation de type DFO sont disponibles sur Internet. Dans un article récent, Moré-Wild [74] effectue un banc d'essais pour comparer les performances de plusieurs algorithmes DFO. On y trouve APPSPACK [38], NMSMAX [44] et NEWUOA [84]. Les résultats de ces tests se limitent aux problèmes d'optimisation sans contrainte. Nous avons expérimenté l'algorithme APPSPACK pour optimiser la fonction δ. Deux éléments propres à la fonction δ réduisent son efficacité : les « régions plates » et les contraintes non-linéaires. Concernant les régions plates, nous avons testé APPSPACH sur des exemples pour lesquels les contraintes sur δ sont uniquement linéaires et les résultats des expériences ont révélé des problèmes de convergence. Plus précisément, nos résultats indiquent que APPSPACH fonctionne bien à condition que la solution de démarrage ne soit pas trop éloignée de l'optimum de la fonction objectif. Dans le cas contraire, le processus de recherche peut présenter un comportement erratique, qui selon nous est dû aux nombreuses régions plates de la fonction δ. Ainsi, la condition pour que APPSPACH fonctionne bien contredit les objectifs de la méthode de calibrage car elle suppose une connaissance a priori du calibrage, c'est-à-dire sur le système. Concernant les contraintes non-linéaires, leur implantation sur APPSPACH est difficile. En effet le procédé courant qui consiste à introduire des pénalités croissantes ou de barrières à mesure qu'on se rapproche des contraintes [78, Chapitre 17] pose problème ici. D'une part ces mécanismes sont difficiles à implanter sur des fonctions

ni continues, ni dérivables. D'autre part, pour certaines contraintes non-linéaires de δ (cf. Section 2.4.2), il est difficile de connaître le degré de violation d'une solution qui les transgresse, c'est-à-dire de quantifier de combien est violée la contrainte concernée. Aussi pour cet ensemble de raisons, nous avons décidé de développer un algorithme type DFO spécialement destiné à ce type de problème.

Contrairement aux routines de cet état de l'art des algorithmes DFO, nous présentons un algorithme assez heuristique, robuste et pragmatique. Il est construit « sur mesure » pour des fonctions objectif δ provenant du calibrage d'un modèle sur un jeu de mesures. Aussi s'adresse-t-il principalement à des fonctions objectif assez régulières et comportant éventuellement de nombreuses régions plates et pour lesquelles des arrondis numériques peuvent biaiser les évaluations de la fonction objectif. Par ailleurs cet algorithme met à profit, comme nous le verrons, certaines spécificités de notre problème de calibrage, notamment concernant les contraintes sur δ.

2.5.2 Notre algorithme DFO

Le fonctionnement général de notre algorithme consiste à démarrer avec une solution possible de calibrage, c'est-à-dire un vecteur possible de paramètres $\beta^0 = (\beta_1^0, \ldots, \beta_n^0)$ et de l'améliorer par des itérations successives jusqu'à obtenir une solution suffisamment bonne. Il n'y a pas d'heuristique générale pour choisir le vecteur de démarrage β^0. On l'engendre aléatoirement dans les plages de valeurs autorisées pour chacun des paramètres. Si le vecteur choisi viole malgré tout une des contraintes(cf. Section 2.4.2, page 33), on répète le processus aléatoire jusqu'à ce qu'un vecteur possible soit trouvé. Généralement, ce simple algorithme de type Las Vegas suffit à trouver rapidement un vecteur possible.

Nous décrivons à présent une étape de notre algorithme itératif. On désigne par β^k le vecteur possible courant. L'idée clé consiste à approcher localement la fonction δ par une quadrique [29]. Le minimum de cette fonction quadratique peut facilement être calculé et constitue une bonne heuristique pour améliorer β^k.

Pour faciliter la compréhension de l'algorithme, un *point* désigne ici un vecteur de paramètres $\beta = (\beta_1, \ldots, \beta_n)$. De plus, un point est qualifié de :

- *vert* s'il satisfait à toutes les contraintes (i.e. il s'agit d'un vecteur possible) ;
- *orange* si la fonction objectif δ peut être calculé mais que le vecteur viole néanmoins une contrainte ;
- *rouge* si le vecteur est en dehors du domaine de définition de δ (soit parce que δ ne peut pas être calculé, soit parce que le modèle associé à ce vecteur correspond à un calibrage irrationnel du modèle).

A chaque itération, il y a trois étapes principales :

1. On sélectionne aléatoirement $p = (n^2 + 3n + 2)/2$ points non-rouges[2] dans le « voisinage » de β^k. La taille de ce voisinage est définie par un vecteur de rayons $\rho = (\rho_1, \ldots, \rho_n)$.
2. On itère aux maximum $t_{\max} = 10$ fois :
   (a) Si un de ces points est vert et améliore la fonction objectif δ, il devient le nouveau point courant β^{k+1} et on retourne à l'étape 1.
   (b) Ces p points engendrent une quadrique unique f. Soit $\beta^\star$ le point critique où les dérivées de f s'annulent. Si $\beta^\star$ est vert et qu'il améliore la fonction objectif δ, il devient le nouveau point courant β^{k+1} et on va à l'étape 1.
   (c) On remplace aléatoirement un des p points par un autre point pris dans le voisinage de β^k.
3. Si cette étape est atteinte, cela signifie que β^k n'a pas été amélioré au cours des $t_{\max}$ itérations de la boucle précédente. On réduit alors le voisinage en diminuant un des rayons ρ_i. Ensuite, on retourne à l'étape 1 jusqu'à satisfaire une des conditions d'arrêt.

Une propriété originale de notre algorithme consiste à tenir compte de ces points oranges qui violent pourtant certaines contraintes. Ils sont utilisés uniquement pour réaliser l'approximation quadratique et semblent être particulièrement utiles lorsque β_{opt}, l'optimum recherché, se situe à la bordure de certaines contraintes.

Dans notre implantation, les p points choisis à l'étape 1, sont sélectionnés aléatoirement à la surface d'une ellipsoïde centrée sur β^k et dont les demi-axes sont de longueur $\rho_1, \ldots, \rho_n$. Ce procédé conduit notre méthode à avancer de façon aléatoire lorsque le point courant β^k se situe sur une région plate, ce qui s'avère être souvent une façon de faire plus efficace que de se fier aux gradients [96, 79].

Lorsque la taille de l'ellipsoïde doit être réduite à l'étape 3, on réduit la taille du rayon d'index i, i.e. ρ_i, pour lequel $\max\{\delta(\beta_1^k, \ldots, \beta_i^k + \rho_i, \ldots, \beta_n^k), \delta(\beta_1^k, \ldots, \beta_i^k - \rho_i, \ldots, \beta_n^k)\}$ est le plus grand. Cette façon de réduire le voisinage vise à rendre l'ellipsoïde plus isotrope. Notons au passage que les rayons ρ_i ne sont jamais augmentés pendant l'exécution de l'algorithme. Enfin, concernant le vecteur ρ, nous conseillons de l'initialiser en tenant compte autant que possible de la signification physique des paramètres, comme nous le verrons plus en détails dans la Section 2.6. Toutefois, ses valeurs initiales ont peu d'impact sur les performances de l'algorithme.

[2] ces p points sont destinés à engendrer la quadrique.

On utilise deux critères d'arrêt dans notre implantation. Le premier intervient lorsque le plus grand des rayons de l'ellipsoïde devient plus petit qu'un certain seuil (typiquement autour de 10^{-4}). Dans ce cas, on considère que la solution courante est une estimation suffisamment précise d'un minimum local de la fonction. Le deuxième critère d'arrêt, caractéristique des algorithmes itératifs, se produit lorsque $\delta(\beta^k)$ n'a pas été amélioré depuis un certain nombre d'itérations (typiquement 10^3).

2.6 Résultats numériques

2.6.1 La fonction objectif

Comme nous l'avons dit dans la Section 2.4 de ce chapitre, le choix de la fonction objectif δ est fonction de la nature des mesures. Pour le cas d'étude A, puisqu'on ne connaît pas les niveaux de charge qui ont engendré les points de mesure, les points couplés C_i sont choisis comme ayant un débit moyen $\bar{X}$ égal à celui des points de mesure M_i. Ensuite, on utilise la métrique multi-critères définie à la Formule (2.8) avec θ à 0.5 et des w_i égaux pour calculer δ. Les contraintes non-triviales[3] auxquelles sont soumis les paramètres du modèle sont les suivantes :

1. $\max_i(\bar{X}_i^{mes}) \leq 1/\tau$
2. $\forall i, \tau \leq \bar{R}_i^{mes} \leq K\tau$
3. $K \geq \max_i(\bar{R}_i^{mes}\bar{X}_i^{mes})$

La première contrainte est due au fait que le débit moyen du modèle ne peut excéder le taux de service du serveur. Si un vecteur de paramètres viole cette contrainte, c'est qu'au moins un M_i est situé au-delà du seuil de saturation du modèle. Il est alors impossible d'associer un point couplé à ce point de mesure M_i (étant donné la définition choisie pour δ). On ne peut donc pas évaluer la valeur de la fonction objectif δ : le vecteur est alors rouge (cf. Section 2.5.2). La deuxième contrainte a déjà été décrite dans la Section 2.4.2. Quant à la troisième contrainte, elle vient de la loi de Little [56]. Contrairement à la contrainte (1), les contraintes (2) et (3) sont oranges. En effet, si un vecteur viole une de ces deux contraintes, la fonction objectif δ peut tout de même être calculée. La relaxation de certaines contraintes, due aux incertitudes intrinsèques aux mesures (cf. Section 2.4.2), ne concerne naturellement que les contraintes oranges. Dans notre cas, on choisit d'augmenter (resp. diminuer) de 10% les limites des contraintes (2) et (3).

[3] il existe également des contraintes triviales comme par exemple la positivité de certains paramètres

Pour le cas d'étude B, le niveau de charge du système est connu pour chacun des points de mesure. Comme conseillé dans la Section 2.4.3, on dérive les points couplés C_i à partir des performances du modèle lorsqu'il est soumis aux mêmes nombres de sources que celui des points de mesure M_i. δ est ensuite calculé en utilisant la métrique multi-critères définie par la Formule (2.6) avec θ à 0.5 et des w_i tous égaux. On note que dans ce cas-là, la métrique Euclidienne et la métrique multi-critères sont équivalentes. Les paramètres du modèle sont sujets à une seule contrainte (non-triviale) :

1. $\forall i, t_s \leq \bar{R}_i^{mes} \leq (S_i - 1)\frac{t_s}{C} + t_s$

La raison de cette contrainte est d'une part que le temps de séjour moyen d'une requête dans le serveur central est évidemment plus grand que le temps moyen nécessaire à la servir. D'autre part, le temps de séjour moyen maximal est atteint lorsqu'une requête entre dans la file alors que les $S_i - 1$ autres requêtes sont déjà en service. Cette contrainte est orange puisque δ reste calculable lorsqu'un vecteur la viole.

2.6.2 Initialisation de l'algorithme DFO

Le vecteur de démarrage β_0 qui doit être vert est engendré par des heuristiques. Pour le cas d'étude A, on garantit que $\beta^0 = (\tau, K)$ est vert en sélectionnant une valeur initiale de K supérieure à $\max_i(\bar{R}_i^{mes}\bar{X}_i^{mes})$ et en choisissant aléatoirement pour τ une valeur comprise entre $\max_i(\bar{R}_i^{mes})/K$ et $\min_i(1/\bar{X}_i^{mes}, \bar{R}_i^{mes})$. Concernant le cas d'étude B, les valeurs initiales de γ, C et t_s sont aléatoirement choisies respectivement dans les intervalles $[10^{-5}, 10]$, $[1, 100]$ et $[\max_i(\bar{R}_i^{mes}\frac{C}{S_i+C-1}), \min_i(\bar{R}_i^{mes})]$. Notons que les deux premiers intervalles ne sont pas reliés à des contraintes du modèle et qu'ils ont été arbitrairement choisis. Cependant ce choix n'est pas crucial puisqu'il ne préjuge pas de la solution finale (i.e. le calibrage trouvé).

L'initialisation de l'algorithme nécessite aussi de fixer la taille initiale du voisinage de β_0, i.e. de dimensionner les demi-axes initiaux ρ_i de l'ellipsoïde. Dans la mesure du possible, on initialise chaque ρ_i de façon à ce qu'il soit inférieur d'un ordre de grandeur à la valeur attendue pour le paramètre associé, β_i. Pour le cas d'étude A, en supposant qu'au moins un des points de mesure correspond à une forte charge en entrée du système, les contraintes (1) et (3) permettent d'estimer l'ordre de grandeur des paramètres β_1 et β_2. On initialise alors ρ_1 à $1/(10 \max_i(\bar{X}_i^{mes}))$ et ρ_2 à $\max_i(\bar{X}_i^{mes}\bar{R}_i^{mes})/10$. Dans le cas d'étude B, on initialise notre algorithme DFO avec $\rho_1 = 0.1$, $\rho_2 = 10$, et $\rho_3 = \min_i(\bar{R}_i^{mes})/10$. Les deux premières valeurs sont arbitrairement choisies tandis que la troisième vient de la borne basse de la contrainte.

2.6.3 Les modèles calibrés par notre méthode

Nous avons calibré les modèles des cas d'étude A et B avec notre méthode de calibrage et nous comparons à présent leur comportement avec les mesures du système. Pour le cas d'étude A, le calibrage obtenu donne les valeurs suivantes pour (τ, K) : (6.95E-03,289.7). Pour le cas d'étude B, les valeurs obtenues par le calibrage pour (γ, C, t_s) sont (1.7, 1.53E02, 6.7E0-3). Les Figures 2.8 et 2.9 représentent les performances des modèles ainsi calibrés et également celles des points de mesure des systèmes concernés. Comme le montrent ces figures, les performances de ces modèles sont très proches des points de mesure ce qui permet de valider à la fois la proposition du modèle et le calibrage obtenu. L'écart moyen entre les points de mesure et les points couplés associés est de 7% pour le cas d'étude A et il est inférieur à 1% pour le cas d'étude B. Notons que ces écarts peuvent être attribués à l'erreur de modélisation (cf. Chapitre Introduction) ou à des imprécisions sur les mesures. Nous présentons dans le Chapitre 5 quelques uns des usages possibles une fois le modèle calibré obtenu.

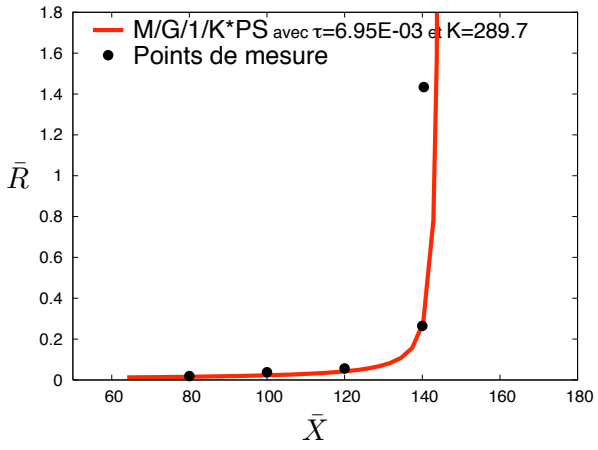


FIG. 2.8 – Le calibrage trouvé pour la file M/G/1/K*PS du cas d'étude A

2.6.4 Evaluation des performances de l'algorithme DFO

Nous avons comparé les performances de notre algorithme DFO avec SolvOpt [50] et avec une approche systématique, encore appelée « brute force". Le temps CPU, le nombre d'évaluations de δ (désigné par $\#\delta$), et le taux de convergence (vers l'optimum), aussi appelé *robustesse* de l'algorithme, sont les trois mesures utilisées pour évaluer les performances de ces trois algorithmes. La robustesse de l'algorithme indique le pourcentage de réalisations de l'algorithme qui convergent effectivement vers le minimum

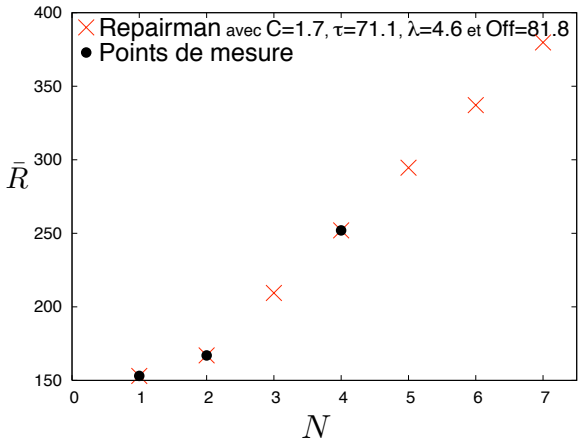


FIG. 2.9 – Le calibrage trouvé pour le modèle « machine repairman » du cas d'étude B

recherché de δ ; le pourcentage restant convergent vers des solutions sous-optimales, correspondant probablement à des minimums locaux de δ. Chaque algorithme est implanté en C et est exécuté sur un processeur Intel Core 2 Duo de 2 GHz. Nous avons répété plusieurs milliers de fois les expériences en faisant varier le vecteur de démarrage β_0. La Table 2.1 montre les performances moyennes obtenues pour ces trois algorithmes sur les cas d'étude A et B.

Cas d'étude	DFO			Solvopt			« Brute force"	
	CPU	$\#\delta$	Robust	CPU	$\#\delta$	Robust	CPU	$\#\delta$
A	359ms	1.42E03	100%	247ms	1.13E03	77%	75s	$1.49E06$
B	11.1s	$3.11E04$	89%	6.2s	1.79E04	23%	> 10h	$1.12E09$

TAB. 2.1 – Les performances de trois algorithmes d'optmisation

Nos résultats indiquent que notre algorithme DFO obtient un taux de convergence plus élevé que SolvOpt pour un surcoût limité de calcul. De plus, notre expérience sur cet algorithme indique que sa complexité (exprimée par $\#\delta$) grandit « raisonnablement » lorsque le nombre de paramètres indéterminés du modèle augmente, contrairement à la complexité combinatoire de l'approche systématique. Il est toutefois difficile de démontrer ce comportement par des résultats expérimentaux, car en changeant le nombre de paramètres à calibrer, on change le problème et a fortiori d'autres facteurs influençant les performances de l'algorithme. Ainsi, nous avons observé que pour des exemples où le nombre de paramètres à calibrer est le même, $\#\delta$ peut passer du simple au triple.

Une analyse rapide de notre algorithme DFO pourrait aboutir au résultat que sa

complexité est polynomiale puisque le nombre de points requis pour générer la quadrique, qui déterminent le nombre d'évaluations de la fonction objectif, augmente en $O(n^2)$ avec le nombre de paramètres, n. Cependant, cette estimation très simple sous-estime très certainement la véritable complexité de notre algorithme car elle ne tient pas compte du fait que lorsque le nombre de paramètres à calibrer augmente, les tentatives de la quadrique échouent plus souvent, ce qui conduit l'algorithme à générer des quadriques supplémentaires.

Cette partie aurait nécessité un travail plus approfondi. Toutefois, malgré sa taille limitée, cette brève étude supporte l'expérience générale que nous avons eu avec notre algorithme DFO, du moins pour les modèles comportant jusqu'à un dizaine de paramètres indéterminés. Notre algorithme apparaît comme un bon compromis entre les méthodes numériques basées sur les gradients qui ne sont pas assez robustes (comme SolvOpt) et les approches systématiques qui deviennent vite intraitables lorsque le nombre de paramètres dépassent 3 ou 4.

2.6.5 Choix des points couplés et performances de l'algorithme DFO

L'impact du choix des points couplés sur l'algorithme de recherche DFO constitue une question intéressante à laquelle il est difficile de répondre. Pour commencer, notons que l'existence de cet impact n'est pas étonnant puisqu'en changeant la sélection des points couplés, on modifie le calcul de δ et donc sa forme et que par conséquent la combinaison des paramètres correspondant à son minimum est certainement déplacée. Il faut distinguer deux aspects à cette question : (1) l'impact sur les performances de l'algorithme de recherche DFO (vitesse de convergence) et (2) l'impact sur le calibrage trouvé, c'est-à-dire sur les paramètres obtenus une fois que l'algorithme DFO a convergé. Concernant le premier aspect, notons d'abord qu'il existe plusieurs façons d'apprécier la vitesse d'un algorithme d'optimisation. Le nombre d'évaluations de la fonction δ et la vitesse de calcul exprimée en temps CPU sont deux façons possibles de faire[4]. D'après notre expérience, pour le nombre d'évaluations de δ, il semble qu'aucune des stratégies présentées pour sélectionner les points couplés ne soit véritablement plus efficace que l'autre. En revanche, la stratégie qui consiste à sélectionner les points couplés avec un paramètre-critère permet d'accélérer l'optimisation en termes de temps CPU car, comme nous l'avons expliqué dans la Section 2.4.3, cette stratégie est généralement plus rapide à exécuter (et plus simple à mettre en oeuvre) ce qui a pour effet de réduire le temps nécessaire à l'évaluation de δ. Concernant le second aspect, relatif à l'impact

[4]Notons que ces deux critères ont été utilisés afin d'évaluer les performances de l'algorithme DFO cf.Tableau 2.1.

du choix des points couplés sur le calibrage trouvé par la technique DFO, nous avons observé qu'à l'exception de quelques rares exemples et qui semblent hélas difficiles à identifier a priori, l'impact sur le calibrage trouvé est généralement limité. Notons pour finir que nous préconisons le choix d'un paramètre-critère pour sélectionner les points couplés de façon à garantir l'homogénéité d'unité dans le calcul de δ.

2.6.6 Comportements abruptes et régions plates

Nous terminons cette section en présentant le principal problème auquel nous avons eu affaire avec notre méthode de calibrage. Il concerne les points couplés, peut se produire lorsque les points de mesure reflètent un comportement très abrupte et favorise la présence de régions plates dans la fonction δ qui peuvent entraver la recherche du calibrage.

Pour illustrer ce problème, nous nous appuyons sur un exemple de jeu de mesures reflétant un comportement très abrupte. La Figure 2.10 décrit cet exemple. La pente entre les deux derniers points de mesure est en effet très raide, ce qui indique que dans le voisinage de ces points, une faible variation de la charge, même très légère, peut provoquer des changements considérables sur $\bar{R}$, et être en revanche sans effet apparent sur $\bar{X}$. Ce type de comportement correspond à une saturation très rapide du système. Par ailleurs, les deux modèles considérés dans cette figure délivrent des performances différentes : les performances du modèle 1 sont incontestablement plus proches des points de mesure que ne le sont celles du modèle 2. Pourtant, les deux modèles sont appréciés (quasi-)pareillement par la fonction δ. Cela tient au fait, que les points couplés choisis pour les deux modèles sont (quasi-)identiques. Notons qu'ici les points couplés ont été choisi avec $\bar{X}$ comme paramètre-critère mais qu'un problème similaire peut se produire s'ils sont sélectionnés comme les points les plus proches (cf. Section 2.4.3). Le résultat de cette situation c'est que la fonction objectif δ peut avoir tendance à niveler ses évaluations lorsque les jeux de mesure présentent des comportements à forte pente.

Par conséquent, les jeux de mesures comportant des pentes très raides favorisent l'apparition de régions plates dans la fonction δ. Or, comme nous l'avons dit précédemment, la présence de régions plates dans la fonction objectif complique la recherche d'un optimum quel que soit l'algorithme de recherche mis en oeuvre. Il n'est donc pas surprenant que nous ayons rencontré plus de difficultés pour ces jeux de mesures aux comportements très abruptes avec des baisses plus ou moins importantes du taux de convergence de notre algorithme de recherche DFO.

Bien que la présence de ces régions plates peut assurément ralentir et compliquer la recherche du calibrage, soulignons que la solution trouvée par la méthode calibrage

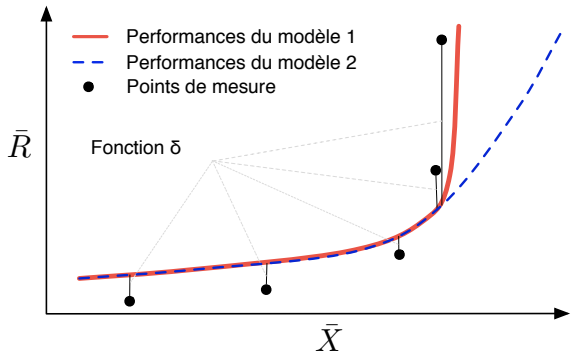


FIG. 2.10 – Un exemple difficile de pente raide

correspond toutefois bien au meilleur calibrage du modèle. Cela tient au fait que les résultats de δ ne sont pas inexacts mais souffrent de précision : des modèles plus proches des points de mesure ne sont pas jugés moins bons que des modèles plus éloignés. Ce détail est important car c'est lui qui garantit que la solution trouvée par la méthode de calibrage correspond effectivement à un bon calibrage du modèle. La Figure 2.11 illustre les difficultés pouvant survenir lorsque les évaluations de δ sont trop nivelées (comme c'est le cas sur la Figure 2.10). La courbe continue de cette figure correspond à un jeu de mesures sans difficulté particulière. Son allure linéaire indique que l'évaluation de δ permet de distinguer sans ambiguïté les calibrages plus proches des mesures de ceux plus éloignés. A l'inverse, la courbe en pointillé décrit une situation plus compliquée puisqu'il apparaît sur cette figure que l'évaluation de δ rend un résultat (quasi-)identique pour toute une plage de calibrages du modèle dont les niveaux de proximité réels aux mesures sont pourtant différents. Les régions plates engendrées compliquent la recherche de la solution, mais comme le montrent certains exemples réels de jeux de mesures au comportement très abrupte dans le chapitre 4 de ce rapport, elles n'empêchent pas la solution de calibrage de fonctionner.

2.7 Conclusion : atouts et limites de cette approche

Nous avons présenté dans ce chapitre une méthode de calibrage pour les modèles de type file d'attente. Les valeurs des paramètres du modèle sont décidées de façon à ce que les performances du modèle approchent au plus près celles reflétées par un jeu de mesures (obtenu sur le système concerné). La recherche de ces paramètres est traitée comme un problème d'optimisation continue ce qui suppose la définition d'une fonction objectif et le développement d'un algorithme de recherche. La méthode de cali-

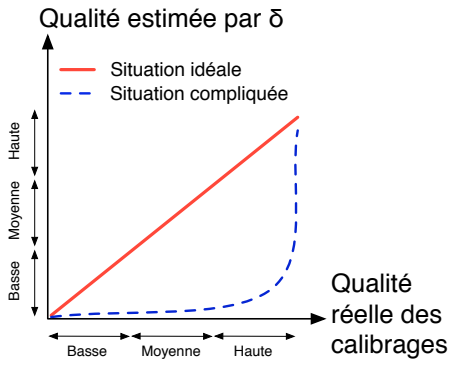


FIG. 2.11 – Comportement de la fonction objectif δ

brage est destinée à des modèles pouvant comporter jusqu'à une dizaine de paramètres indéterminés.

Chacun des aspects de cette méthode de calibrage a été pensé pour être le plus général possible, simple d'utilisation et applicable automatiquement. Pour la fonction objectif δ, dont le rôle est d'évaluer la qualité d'un calibrage donné du modèle en quantifiant son éloignement aux points de mesure, nous avons proposé des définitions simples à mettre en oeuvre et applicables à des modèles et jeux de mesures de nature variée. Concernant, l'algorithme de recherche, nous avons développé un algorithme type DFO, qui ne fait aucune hypothèse sur la fonction objectif, et a fortiori sur des propriétés du modèle. Pour fonctionner, cet algorithme ne requiert que l'évaluation de la fonction objectif δ, ce qui le rend utilisable pour la plupart des modèles, y compris ceux résolus par méthode numérique ou par simulation[5]. Cet algorithme, qui repose sur une approximation quadratique de la fonction objectif, s'accommode simplement et efficacement de contraintes non-linéaires ce qui représente un avantage décisif puisque la plupart des modèles sont soumis à de telles contraintes. A travers deux cas d'étude dérivés de situations réelles, nous avons illustré étape par étape le fonctionnement de cette méthode de calibrage. Enfin, nous avons comparé, sur ces deux cas d'étude, les performances de l'algorithme DFO avec d'autres algorithmes existants. Les résultats obtenus indiquent que l'algorithme DFO est sensiblement plus robuste que les autres, moyennant un temps de recherche légèrement plus long, ce qui corrobore l'expérience générale que nous en avons eue.

La vitesse d'exécution de la méthode de calibrage dépend directement du temps nécessaire pour évaluer la fonction objectif δ, et a fortiori du temps de résolution du modèle considéré. Pour des modèles à 3 ou 4 paramètres indéterminés pour lesquels

[5]Rappelons que la durée d'une simulation peut être extrêmement longue.

une solution analytique efficace existe (par exemple, une file M/M/C/K ou un modèle « machine repairman »), le calibrage est trouvé en moins de quelques secondes[6]. En revanche, lorsque le nombre de paramètres indéterminés dépasse 8 ou 9, notre algorithme DFO atteint ses limites et la recherche du calibrage peut prendre plusieurs dizaines de minutes. Pour ces cas-là, on peut envisager de conserver le cadre général établi dans ce chapitre pour la recherche et opter pour un algorithme DFO plus sophistiqué dont le rendement se renforce à mesure que le nombre de variables inconnues augmente [85].

Nous concluons ce chapitre consacré au calibrage automatique de modèles par quelques remarques concernant le caractère automatique de notre méthode car elle constitue son originalité majeure. Premièrement, telle que nous l'avons défini, le calcul de la fonction objectif δ est systématique. Lorsque les niveaux de charge des mesures ne sont pas connus, la procédure permettant d'inférer automatiquement les points couplés nécessaires au calcul de δ est simplifiée si les points couplés sont sélectionnés selon un paramètre-critère. Si les niveaux de charge des mesures sont connus, le calcul de δ est alors direct. Deuxièmement, notre algorithme de recherche DFO ne fait aucune hypothèse sur la fonction objectif δ et peut s'exécuter automatiquement dès lors qu'on peut évaluer δ. C'est la combinaison de ces deux éléments qui permet à notre méthode de calibrage de fonctionner pour des modèles et des jeux de mesures de nature variée tout en étant automatique.

[6] avec un processeur Intel Core 2 Duo de 2 GHz

Chapitre 3

HLM - une nouvelle approche de modélisation

Sommaire

3.1 Etat de l'art

A notre connaissance très peu de travaux ont été consacrés à la génération automatique de modèles pour des systèmes informatiques. En informatique, comme dans la plupart des secteurs, le processus de modélisation d'un système repose généralement sur une approche constructive[1], ce qui rend très difficile si ce n'est impossible son automatisation.

Il nous paraît cependant utile de présenter des méthodes d'inférences qui s'apparentent par certains aspects à la méthode de modélisation que nous décrivons dans ce chapitre. Le principe général d'une *méthode d'inférence* consiste à observer le système et à rechercher ensuite grâce à un modèle présupposé du système les causes qui ont produit ces effets. En pratique, cela revient souvent à interpréter le sens de certaines mesures prises sur un système afin de déterminer quel était l'état de ce système aux instant des mesures. Ainsi, les méthodes d'inférence ont en commun avec notre méthode de modélisation de s'appuyer sur des mesures, d'estimer les performances du système, et pour certaines d'entre elles, de pouvoir s'exécuter automatiquement.

Un exemple typique d'application des méthodes d'inférence concerne le calcul de la taille transitoire d'une file d'attente en connaissant uniquement les instants de débuts et de fins de service de chaque client (i.e. les données transactionnelles) [61, 11, 31, 32]. Ces méthodes s'adressent donc aux systèmes dont le fonctionnement est assimilable à une file d'attente et pour lesquels on connaît les données transactionnelles mais pas directement la taille de leur file. Les guichets automatiques bancaires des agences, les systèmes de radio mobile, les cabines téléphoniques publiques, les caisses de supermarchés, et certains noeuds des réseaux informatiques sont les principaux systèmes auxquels sont destinées ces méthodes. Larson a été le premier à développer une méthode pour déduire le comportement d'une file uniquement à partir de ses données transactionnelles [61]. Son algorithme repose sur l'hypothèse que le processus d'arrivées des clients dans la file est poissonnien, ce qui signifie que le modèle d'inférence considéré ici est une file multiserveur à arrivées poissoniennes. Bertsimas et Servi [11], puis Daley et Servi [31] et également Dimitrijevic [32] ont apporté des changements à l'algorithme de Larson pour permettre notamment un calcul plus efficace de la taille de la file, une estimation du délai d'attente pour chacun des clients et également une estimation de la taille la plus probable de la file (qui dans certains cas peut s'écarter sensiblement de sa taille moyenne). En pratique, du moins lorsque le coefficient de variation des temps de service de la file est inférieur à 1, les inférences délivrées par ces algorithmes fonc-

[1] Cette approche a été décrite dans le Chapitre Introduction.

tionnent assez bien. Leur calcul relativement simple et rapide permet d'automatiser ces méthodes et même de les utiliser en ligne pour calculer dynamiquement les tailles transitoires de la file [11].

Un autre exemple de méthodes d'inférence, relatif cette fois-ci aux réseaux informatiques, concerne les chemins réseaux sur Internet. Alouf et al. [3] puis Salamatian [90] ont développé des méthodes d'inférence leur permettant d'estimer certaines propriétés du chemin entre deux hôtes sur un réseau à partir de mesures prises sur le chemin de bout-en-bout. Alouf et al. ont notamment pu dériver des formules analytiques permettant d'estimer la taille du buffer et l'intensité du « cross traffic » au niveau du lien goulet d'étranglement du chemin considéré. Dans cet exemple, le modèle d'inférence choisi par les auteurs est une simple file à capacité d'accueil finie.

Ainsi, les méthodes d'inférence peuvent permettre de déterminer l'état d'un système en faisant appel à des mesures partielles prises sur le système à cet instant. Toutefois, ces procédés ont certaines limites. Premièrement, l'intérêt de ces méthodes réside principalement dans la description et la compréhension du système (en estimant par exemple la valeur de certaines quantités) ; elles ne sont pas pensées pour prévoir les performances à venir du système. Deuxièmement, ces méthodes d'inférence sont très spécifiques et semblent difficilement généralisables à d'autres systèmes car les raisonnements et les calculs engagés dépendent étroitement du modèle présupposé pour le système. Troisièmement, ces méthodes s'adressent aux systèmes pour lesquels on dispose d'un modèle immédiat, i.e. le fonctionnement du système suggère un choix naturel de modèle (comme dans le premier exemple), ou pour lesquels on présuppose un modèle (comme pour le second exemple). Or découvrir un modèle adéquat pour un système est généralement un exercice difficile (cf. Chapitre Introduction), surtout si la connaissance du système se limite essentiellement à des mesures. Notons que c'est précisément l'objectif de notre méthode que de générer un modèle simple à partir des mesures du système.

A présent, considérons le problème suivant : modéliser par une file d'attente le comportement d'un système dont on ignore le fonctionnement interne. De fait, l'approche constructive apparaît comme inadaptée pour décider du choix de la file car sa mise en oeuvre suppose une connaissance intrusive du système. Cependant, l'observation du comportement du système peut permettre de guider le choix de la file à utiliser. Nous présentons deux exemples. Premièrement, contrairement à une file avec un seul serveur, le taux de service d'une file multiserveur augmente avec le nombre de clients dans la file (jusqu'à ce que tous les serveurs soient occupés). Supposons qu'on parvienne à observer le comportement du système lorsque n clients entrent simultanément dans le système à vide. Si l'on observe que jusqu'à $n \leq C$, les n clients quittent en moyenne le système

à des instants identiques après un temps t_0, alors la file modélisant le système devra certainement être une file multiserveur avec C serveurs dont les temps de service moyen sont égaux à t_0 [47]. Deuxièmement, il est connu que plus le nombre de serveurs d'une file est élevé, plus la saturation de la file démarre proche de sa capacité maximale de traitement : la saturation intervient plus tardivement et plus vite [56]. En observant la vitesse de saturation d'un système grâce à un ensemble de mesures pris à différents niveaux de charge, on peut ainsi avoir une indication quant au nombre de serveurs que doit avoir la file recherchée. Bien qu'elles permettent d'orienter le choix d'un modèle pour des systèmes vus comme des boîtes noires, les déductions de ce type sont évidemment insuffisantes pour décider pleinement du choix d'un modèle. L'objectif de notre méthode est de savoir si l'on peut aller plus loin, c'est à dire de décider pleinement du choix d'un modèle en se basant uniquement sur des observations du système.

Nous pensons, et c'est l'hypothèse sous-jacente à ce travail, que l'observation d'un système de type boîte noire, combinée à une méthode restant à définir, peut aboutir à des résultats de modélisation plus significatifs que les deux exemples présentés ci-dessus. Plus précisément, ce travail vise à définir une méthode de modélisation qui repose entièrement sur les mesures pour déterminer un modèle. Pour y parvenir, nous pensons qu'il faut disposer de plusieurs points de mesures correspondant à des niveaux de charge différents du système, c'est-à-dire disposer d'un jeu de mesures (cf. Chapitre 2). Selon nous, un ensemble de plusieurs points de mesure peut caractériser le comportement d'un certain nombre de systèmes et, à condition d'avoir une méthode qui permet de le révéler, peut suffire alors à fournir un modèle, du moins dans un certain nombre de cas. C'est précisément l'objectif de notre méthode qui est de se servir d'un ensemble de points de mesures, pris à des niveaux de charge différents du système, pour découvrir un modèle qui reproduit le comportement mesuré du système.

Au passage, notons que puisque les jeux de mesures considérés dans ce chapitre sont composés de points de mesure pris à des niveaux de charge différents du système, cela rend de fait inadaptées les méthodes qui cherchent à ajuster un modèle de distribution discrète ou continu à un échantillon de mesures obtenues pour un même niveau de charge [21].

Pour terminer, il peut être tentant lorsqu'on dispose d'un jeu de mesures de faire appel à une méthode d'interpolation pour modéliser le comportement du système entre les points de mesure plutôt que de chercher un modèle de type file d'attente (quelle que soit d'ailleurs la méthode empruntée). D'après notre expérience, ce type d'approche par interpolation a très peu de chance de réussir. En effet, notre méthode de modélisation utilise des modèles de type file d'attente dans lesquels, à l'instar du système

étudié, les clients subissent des phénomènes de congestion et des mécanismes d'attente avant d'être servis. En revanche, comme l'illustre la Figure 3.1, bien qu'une technique d'interpolation basée sur des fonctions mathématiques (interpolations polynomiales, lagrangienne ou avec les fonctions de Bessel par exemple) parvienne certainement à relier tous les points de mesure, elle risque de présenter des comportement irrationnels entre ces points (des valeurs négatives ou infaisables ou encore des oscillations). C'est pourquoi nous pensons que l'ajustement d'une fonction donnée sur des points de mesures est bien moins profitable pour la compréhension du comportement d'un système et pour la prévision de ses performances qu'une méthode basée sur des modèles de congestion.

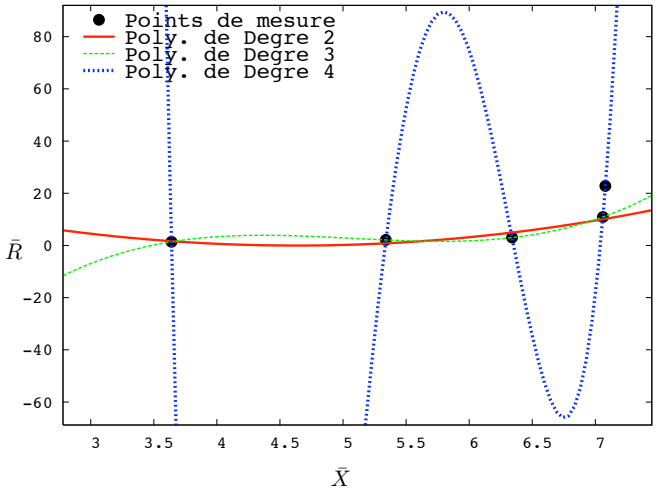


FIG. 3.1 – Des résultats irrationnels pour des interpolations polynomiales

3.2 La méthode HLM

Nous décrivons à présent le fonctionnement général de notre méthode de modélisation nommée *HLM* (pour « High Level Modeling »). La méthode HLM repose sur l'hypothèse qu'il existe pour le système étudié un modèle simple qu'elle cherche à découvrir et à calibrer.

La méthode HLM a les mêmes objectifs qu'une méthode de modélisation ordinaire : elle vise à définir un modèle pour un système donné. Le modèle produit doit pouvoir permettre de mieux décrire, comprendre ou prévoir le comportement du système (cf. Chapitre Introduction).

La méthode HLM ne nécessite qu'un jeu de mesures pour fonctionner. Tout le processus de modélisation, c'est à dire le choix du modèle (et de son environnement) et

son calibrage, doivent être décidés à partir de cette seule connaissance. Contrairement à l'approche constructive, le modèle recherché ne vise pas à reproduire le fonctionnement interne du système considéré mais uniquement ses performances d'entrée/sortie. Autrement dit, le système concerné peut être vu comme une boîte noire pour laquelle on dispose d'un jeu de mesures d'entrée/sortie. La méthode de modélisation fonctionne si elle parvient à trouver un modèle simple de type file d'attente qui reproduit les performances du système exprimées à travers des mesures, et plus généralement si la file trouvée constitue un modèle possible du système.

Les systèmes auxquels est destinée notre méthode de modélisation sont multiples et variés. Il peut s'agit de processeurs, de contrôleurs disques, de serveurs Web, de réseaux Ethernet, de réseaux sans-fil, de chemins sur un réseau maillé sans fil (cf. Chapitre 4) et plus généralement de pratiquement n'importe quel système informatique pourvu qu'on dispose d'un jeu de mesures. Cette diversité est rendue possible du fait qu'aucune hypothèse n'est faite sur le fonctionnement interne du système étudié. Les systèmes sont traités comme s'il s'agissait de boîtes noires dans lesquelles des clients entrent, y demeurent un certain temps (pouvant comprendre de l'attente en plus du temps de service) et puis en sortent.

Afin de rendre la méthode HLM attractive au plus grand nombre, nous l'avons définie de façon à être aussi générale, simple et automatique que possible. Autrement dit, la méthode doit pouvoir fonctionner pour des jeux de mesures et des systèmes de nature variée et être exploitable par des utilisateurs non spécialistes.

Le principe de la méthode HLM s'articule autour de trois étapes principales. La première étape concerne la définition d'un ensemble de briques. Une *brique* représente un modèle générique (et son environnement) pour lequel certains des paramètres ne sont pas déterminés. Dans ce rapport, l'ensemble de briques considéré inclut des modèles simples issus de la théorie des files d'attente (comme par exemple les files M/M/C et M/G/1) mais également des modèles plus originaux (comme par exemple les files imbriquées). Ensemble, ces briques semblent permettre de couvrir un spectre assez large des comportements observés sur les systèmes et réseaux informatiques. La deuxième étape de notre méthode est le calibrage de chacune de ces briques en fonction du jeu de mesures issu du système. Cette étape vise à déterminer pour chacune des briques la combinaison des paramètres qui la conduit à s'approcher au plus près des points de mesure. Pour réaliser cette étape, on fait appel à la méthode de calibrage automatique qui comprend un algorithme original de type DFO (« Derivative Free Optimization ») et qui a été présentée dans le Chapitre 2 de ce rapport. Enfin, la troisième et dernière étape de notre méthode est la sélection du modèle lauréat. Le modèle lauréat de notre

méthode correspond simplement à la brique calibrée qui reproduit au mieux le jeu de mesures. La Figure 3.2 résume de façon graphique le fonctionnement général de la méthode HLM.

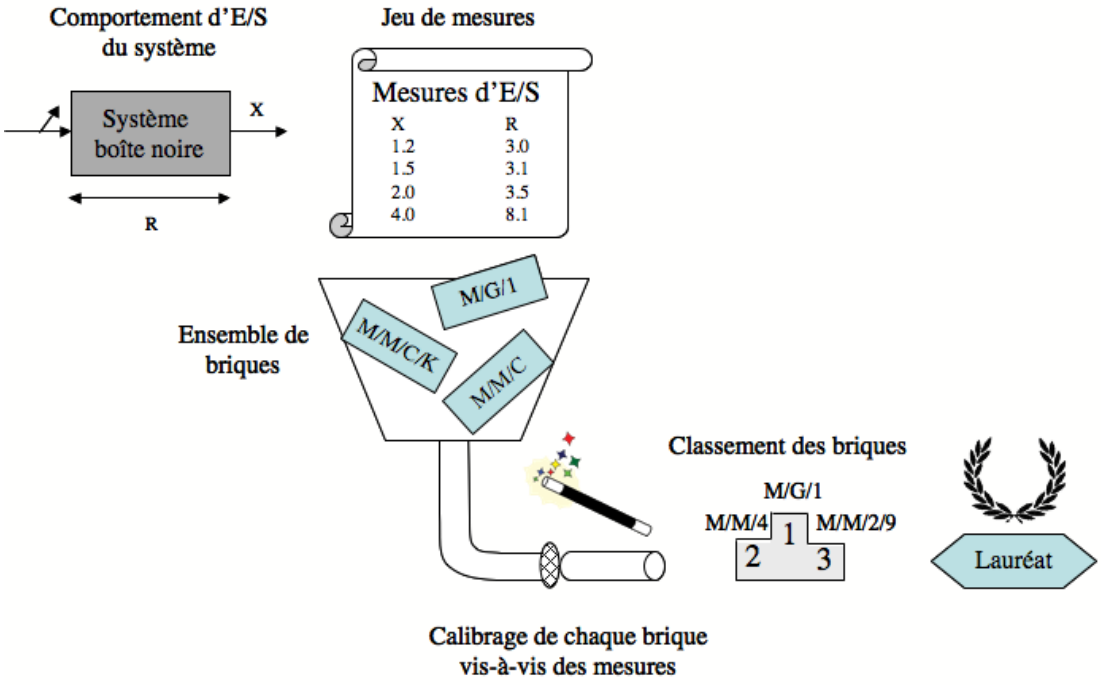


FIG. 3.2 – Le fonctionnement général de la méthode HLM

3.2.1 Les paramètres des briques

Nous présentons dans cette section les modèles génériques qui constituent l'ensemble des briques de notre méthode. Comme nous l'avons dit précédemment, les briques représentent des modèles de type file d'attente (avec leur environnement) dont les paramètres ne sont pas déterminés.

Les paramètres d'une brique peuvent être classés en deux catégories : ceux reliés au modèle et ceux reliés à son environnement. Les premiers que l'on désigne comme les *paramètres structurels* du modèle caractérisent le modèle (sans son environnement). La plupart des paramètres structurels d'un modèle ont une valeur constante (par exemple la capacité d'accueil d'une file). La valeur de certains peut varier mais c'est alors en réponse à une loi inhérente au modèle. Par exemple, pour certains modèles le nombre de serveurs d'une file peut changer lorsque le taux d'arrivée des clients dans la file dépasse un certain seuil. Les paramètres de cette loi, qui incluent entre autres la valeur de ce

seuil, font alors partie des paramètres structurels du modèle. Replacés dans le contexte du Chapitre 2 pour lequel nous avons considéré le calibrage d'un modèle présupposé de type file d'attente, les paramètres structurels de la brique sont des paramètres indéterminés $(\beta_1, \ldots, \beta_n)$ puisque par définition leur valeur est initialement inconnue : elle est décidée en fonction des mesures. Ainsi, sauf à supposer une connaissance supplémentaire sur le système, il n'y a pas de paramètres déterminés $(\alpha_1, \ldots, \alpha_n)$ dans nos briques.

Les *paramètres de charge* constituent la deuxième catégorie des paramètres d'une brique. Leurs valeurs caractérisent l'environnement du modèle et a fortiori les conditions de charge du modèle. Selon les cas, ces paramètres peuvent par exemple correspondre au nombre de sources émettant des clients vers le modèle, à l'intensité de chacune de ces sources ou encore à la taille (exprimée par exemple en durée de temps de service) des clients émis par les sources. C'est en modifiant la valeur de ces paramètres qu'on modifie le niveau de charge du modèle et par conséquent ses conditions de fonctionnement. Les paramètres de charge d'une brique sont à mettre en relation avec les jeux de mesures considérés par la méthode HLM. Rappelons que dans un jeu de mesures, chaque point (de mesure) correspond à un certain niveau de charge, connu ou pas, du système. Or puisque notre méthode recherche un modèle qui reproduit au mieux cette description comportementale du système, elle doit également considérer le comportement du modèle à plusieurs niveaux de charge. C'est pourquoi contrairement aux paramètres structurels dont la valeur est constante ou résulte d'une loi inhérente au modèle, les valeurs des paramètres de charge du modèle, ou du moins certains d'entre eux, doivent varier de façon à pouvoir évaluer le modèle à différents niveaux de charge. Ceux restants, c'est-à-dire les paramètres de charge qui restent fixes d'un point de mesure à un autre, font partie pour la méthode de calibrage présentée dans le Chapitre 2 des paramètres indéterminés $(\beta_1, \ldots, \beta_n)$.

3.2.2 Les briques élémentaires

L'ensemble de briques que nous considérons dans notre méthode contient une sélection de modèles génériques élémentaires issus de la théorie des files d'attente. Nous présentons succinctement ces quelques briques bien connues [56, 2, 5].

La file M/M/C

Il s'agit d'un modèle ouvert formé d'une file FIFO et de plusieurs serveurs. Les C serveurs sont identiques et indépendants les uns des autres. La capacité d'accueil de

la file est illimitée. Les clients sont émis depuis une seule source et leurs arrivées dans la file suivent un processus de Poisson de taux λ. Le temps de service des clients dans chacun des serveurs est distribué selon une loi exponentielle de moyenne $1/\mu$. On définit $\rho = \lambda/\mu$. Soit $p(n)$ la probabilité stationnaire d'avoir n clients dans la file, on a alors :

$$\begin{cases} p(n) = \begin{cases} p(0)\dfrac{\rho^n}{n!} & \text{si } n \leq C \\ p(0)\dfrac{\rho^n}{C!C^{n-C}} & \text{si } n > C \end{cases} \\ \\ \text{avec } p(0) \text{ tel que } \displaystyle\sum_{n=0}^{\infty} p(n) = 1 \end{cases}$$

On en déduit les expressions analytiques suivantes pour le débit moyen de clients en sortie de la file $\bar{X}$, pour le temps de séjour moyen d'un client dans la file $\bar{R}$ et pour le nombre moyen de clients dans la file $\bar{Q}$.

$$\begin{aligned} \bar{X} &= \lambda \\ \bar{R} &= \frac{1}{\mu} + \sum_{n=C}^{\infty} p(n)\frac{n+1-C}{C\mu} \\ \bar{Q} &= \rho + p(0)\frac{\rho^{C+1}}{(C-1)!(C-\rho)^2} \end{aligned}$$

La file M/M/C possède deux paramètres structurels : C et μ et un paramètre de charge : λ. Le calibrage de cette brique consiste à déterminer les valeurs de C et μ de façon à ce que le comportement de la file reproduise au mieux celui décrit par les mesures. Ainsi, lorsqu'on compare un calibrage de la file M/M/C à un jeu de mesures, une valeur différente de λ doit être considérée pour chacun des points de mesure (tandis que les valeurs de C et μ elles restent fixes).

Il existe quelques propriétés remarquables pour les paramètres de performances de cette file. Ces propriétés sont exploitées lors de l'étape de calibrage de la file que nous présentons dans la section suivante afin d'assurer la cohérence du modèle avec les mesures du système.

1. $\bar{X} < C\mu$: le débit moyen en sortie de la file ne peut pas être supérieur à la capacité maximale de traitement de la file ;
2. $1/\mu \leq \bar{R}$: le temps de séjour d'un client comprend au moins son temps de service.

La file M/M/C/K

Une file M/M/C/K est une file multiserveur M/M/C mais avec une capacité d'accueil limitée à K clients. La capacité d'accueil K indique le nombre maximal de clients pouvant être présents au même instant dans la file. Lorsque la file est pleine, aucun nouveau client ne peut entrer dans la file : toute nouvelle arrivée est rejetée. Pour le reste, une file M/M/C/K est identique à une file M/M/C. On a pour $p(n)$ i.e. la probabilité stationnaire d'avoir n clients dans la file :

$$\begin{cases} p(n) = \begin{cases} p(0)\dfrac{\rho^n}{n!} & \text{si } n \leq C \\ p(0)\dfrac{\rho^n}{C!C^{n-C}} & \text{si } C < n \leq K \end{cases} \\ \\ \text{avec } p(0) \text{ tel que } \displaystyle\sum_{n=0}^{K} p(n) = 1 \end{cases}$$

Ainsi, on peut obtenir les formules suivantes pour les paramètres de performances $\bar{X}$, $\bar{R}$ et $\bar{Q}$:

$$\begin{aligned} \bar{X} &= \lambda(1 - p(K)) \\ \bar{R} &= \frac{1}{\mu} + \sum_{n=C}^{K-1} \frac{p(n)}{1 - p(K)} \frac{n + 1 - C}{C\mu} \\ \bar{Q} &= \bar{R}\bar{X} \qquad \text{d'après la loi de Little [65]} \end{aligned}$$

La file M/M/C/K possède trois paramètres structurels : C, K et μ et un paramètre de charge : λ.

Voici quelques propriétés remarquables concernant les paramètres de performances de cette file :

1. $\bar{X} < C\mu$: le débit moyen en sortie de la file ne peut pas être supérieur à la capacité maximale de traitement de la file ;
2. $1/\mu \leq \bar{R} \leq K/C\mu$: le temps de séjour d'un client comprend au moins son temps de service et atteint sa valeur maximale si la file contient $K - 1$ clients lorsque le client entre dans la file ;
3. $\bar{Q} \leq K$: le nombre de clients dans la file ne peut pas être supérieur à la capacité d'accueil de la file.

La file M/G/1

Il s'agit d'un modèle ouvert formé d'une file FIFO et d'un seul serveur. La capacité d'accueil de la file est illimitée. Les clients sont émis depuis une seule source et leurs arrivées dans la file suivent un processus de Poisson de taux λ. Le temps de service d'un client est distribué selon une variable aléatoire générale de moyenne $1/\mu$ et de coefficient de variation γ. On définit $\rho = \lambda/\mu$. Plusieurs méthodes permettent de calculer les probabilités stationnaires d'avoir n clients dans la file (par exemple, la méthode de la chaîne de Markov incluse [56, 5] ou bien des méthodes (semi-)numériques [20]). Toutefois si seules les moyennes des paramètres de performances $\bar{X}$, $\bar{R}$ et $\bar{Q}$ nous intéressent, il est inutile d'obtenir au préalable les probabilités stationnaires. Grâce à une expression concise du temps de service résiduel d'un client en cours de service [56, 2, 5], on obtient les expressions suivantes connues sous le nom des formules de Pollaczek-Khintchine :

$$\begin{aligned} \bar{R} &= \frac{1}{\mu} + \frac{\lambda(1+\gamma^2)}{2\mu^2(1-\rho)} \\ \bar{X} &= \lambda \\ \bar{Q} &= \rho + \frac{\rho^2(1+\gamma^2)}{2(1-\rho)} \end{aligned}$$

La file M/G/1 possède deux paramètres structurels : μ et γ et un paramètre de charge : λ.

Il existe également quelques propriétés remarquables pour les paramètres de performances de cette file :

1. $\bar{X} < \mu$: le débit moyen en sortie de la file ne peut pas être supérieur au taux de service de la file ;
2. $1/\mu \leq \bar{R}$: le temps de séjour d'un client comprend au moins son temps de service.

Notons ici que nous avons envisagé de faire figurer la file M/G/1/K comme une brique supplémentaire de la méthode HLM. La présence de cette file à capacité d'accueil limitée permettrait d'étendre les possibilités de la méthode en particulier lorsque le jeu de mesures considéré contient des taux de pertes. Hélas, son insertion dans la méthode HLM poserait problème car, comme nous l'explicitons dans le Chapitre 5, il n'existe à notre connaissance aucune caractérisation simple du temps de service qui permette de déterminer sans ambiguïté les paramètres de performance moyens d'une file M/G/1/K. En effet, bien que les probabilités stationnaires de cette file puissent être obtenues à partir de celles d'une file M/G/1, ses paramètres de performances moyens ne dépendent pas seulement des deux premiers moments du temps de service, comme c'est le cas pour

la file M/G/1 [2], ni même des trois premiers [18]. Notons que c'est pour le même type de raisons que nous n'avons pas fait figurer la file M/G/C dans l'ensemble des briques de la méthode HLM.

La file M/M/C//N

Cette brique correspond à une file FIFO multiserveur pour laquelle l'environnement est composé de plusieurs sources. On désigne par N le nombre de sources. Le nombre de clients réparti entre la file et l'environnement est constant et égal à N. Ce modèle représente donc un modèle fermé. Le temps passé par un client dans une source avant d'entrer dans la file est distribué exponentiellement avec un taux λ. La file possède C serveurs identiques et indépendants. Les temps de service des clients dans la file suivent une loi exponentiel de moyenne $1/\mu$. La capacité d'accueil de cette file est illimitée. En régime permanent, le nombre de clients présents dans la file peut se modéliser par un processus de naissance et de mort. Nous avons représenté dans la Figure 3.3 la chaîne de Markov qui correspond à ce processus.

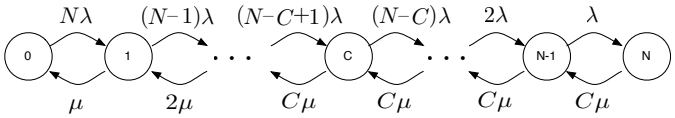


FIG. 3.3 – La chaîne de Markov associé au modèle fermé de type multiserveur

En définissant $\rho = \lambda/\mu$ et en supposant[2] que $N > C$, on obtient pour $p(n)$ i.e. la probabilité stationnaire d'avoir n clients dans la file :

$$\begin{cases} p(n) = \begin{cases} p(0)\dfrac{\rho^n}{n!}\dfrac{N!}{(N-n)!} & \text{si } n \leq C \\ p(0)\dfrac{\rho^n}{C!C^{n-C}}\dfrac{N!}{(N-n)!} & \text{si } C < n \leq N \end{cases} \\ \\ \text{avec } p(0) \text{ tel que } \displaystyle\sum_{n=0}^{N} p(n) = 1 \end{cases}$$

Ainsi, on peut calculer les paramètres de performances $\bar{X}$, $\bar{R}$ et $\bar{Q}$ de cette file de

[2]dans le cas contraire, les clients ne subissent jamais d'attente avant d'être servis

la façon suivante :

$$\begin{aligned} \bar{Q} &= \sum_{n=1}^{N} p(n)n \\ \bar{X} &= (N - \bar{Q})\lambda \\ \bar{R} &= \bar{Q}/\bar{X} \qquad \text{d'après la loi de Little [65]} \end{aligned}$$

Ce modèle fermé possède deux paramètres structurels : C et μ et deux paramètres de charge : λ et N. Par conséquent, lorsqu'on souhaite comparer les performances de ce modèle avec les points de mesure correspondant à plusieurs niveaux de charge sur le système, on peut décider de faire varier N ou bien λ, voire les deux. Notons que si le jeu de mesures indique le niveau de charge correspondant à chaque point de mesure, le choix du paramètre de charge à faire varier est certainement immédiat.

Voici quelques propriétés remarquables pour les paramètres de performances de cette file :

1. $\bar{X} < C\mu$: le débit moyen en sortie de la file ne peut pas être supérieur à la capacité maximale de traitement de la file ;
2. $1/\mu \leq \bar{R} \leq N/C\mu$: le temps de séjour d'un client comprend au moins son temps de service et atteint sa valeur maximale si la file contient $N - 1$ clients lorsqu'il entre dans la file ;
3. $\bar{Q} \leq N$: le nombre de clients dans la file ne peut pas être supérieur au nombre total de clients réparti entre le modèle et son environnement.

3.2.3 L'offest

Afin de représenter le fait que pour certains systèmes le temps de séjour d'un client comprend une partie fixe et incompressible qui ne dépend pas de la charge, nous introduisons un paramètre supplémentaire pour toutes les briques de la méthode : l'*offset*. Nous désignons ce paramètre par *Off* sur toutes les figures et les formules de ce rapport. L'offset correspond simplement à une constante positive qui s'ajoute au temps de séjour moyen évalué par une file :

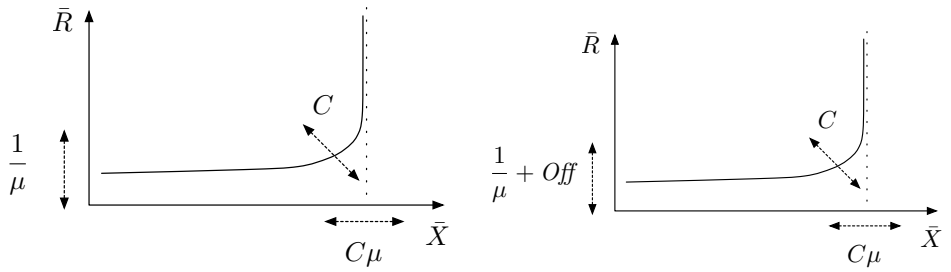
$$\bar{R}^* = \bar{R} + Off$$

où $\bar{R}^*$ représente le temps de séjour moyen d'une brique en présence d'un offset. Il est absolument sans effet sur la congestion de la file. Pour les contrôleurs disques, l'offset peut représenter le temps nécessaire pour examiner une commande E/S avant de la mettre en attente ou de la servir. Pour un serveur d'applications, il peut correspondre

au temps dont a besoin une requête pour parcourir le chemin du poste émettant la requête jusqu'au serveur, à condition que ce temps n'augmente pas si l'intensité de la charge augmente. Pour la transmission de données, la valeur de l'offset peut refléter le délai de propagation du signal entre les noeuds communicants.

Off constitue un paramètre structurel supplémentaire à toutes les briques que nous présentons dans ce rapport. Il peut être représenté par une file M/D/∞ placée en série derrière le modèle principal et pour laquelle le taux de service des serveurs est égal à $1/Off$. La présence de ce paramètre dans toutes les briques permet à l'ensemble des briques de reproduire un spectre significativement plus large des comportements observés des systèmes informatiques.

Pour illustrer le gain apporté par la présence du paramètre *Off*, nous considérons le cas d'une brique M/M/C. Si l'on résume l'influence des paramètres C et μ d'une file M/M/C sur ses paramètres de performances, on a trois caractéristiques essentielles correspondant aux trois degrés de liberté comme l'illustre la Figure 3.4(a) : (1) le produit $C\mu$ détermine le seuil de saturation de la file, (2) le quotient $1/\mu$ correspond approximativement au temps de séjour moyen de la file à faible charge et (3) C détermine la vitesse de saturation de la courbe, c'est-à-dire son rayon de courbure. Or par définition, la brique M/M/C dans sa forme initiale ne dispose que de deux paramètres. Par conséquent, dès lors que deux de ces points essentiels sont fixés, le troisième est automatiquement déterminé. La présence du paramètre supplémentaire *Off* permet de déterminer chacun de ces trois points essentiels indépendamment les uns des autres ce qui se traduit naturellement par une augmentation très significative des possibilités de calibrage du modèle. L'influence d'un Offset sur les performances d'une file M/M/C est représentée graphiquement sur la Figure 3.4(b).



(a) Sans Offset (b) Avec Offset

FIG. 3.4 – Illustration des performances atteignables par une file M/M/C

3.2.4 Des briques plus originales

Nous présentons à présent des briques plus originales qui font également partie de l'ensemble des briques de notre méthode.

Les modèles imbriqués

Les *modèles imbriqués* représentent un ensemble de briques pour lesquelles le temps de service dans la file « principale » comprend le temps d'attente dans une file « secondaire ». La file principale qui est celle dont on mesure les performances est une file M/G/1 dont le temps de service est la somme de deux variables aléatoires indépendantes : le temps de service de base T_b et l'élongation du temps de service T_s. Cette dernière variable est obtenue à partir du temps d'attente W_s dans la file secondaire qui est une file M/G/C. Le taux des arrivées dans la file secondaire est un multiple de facteur η de celui dans la file principale. On désigne par λ le taux d'arrivée des clients dans la file principale et par λ_s le taux d'arrivée des clients dans la file secondaire. On a donc $\lambda_s = \eta\lambda$. La Figure 3.6 illustre le fonctionnement général d'un modèle imbriqué.

Les modèles imbriqués ont été pensés pour des jeux de mesures décrivant des comportements impossibles à reproduire par nos briques élémentaires, ni d'ailleurs semblent-ils par les files pourtant très générales de type M/G/C. Contrairement à toutes ces files, le temps de séjour moyen d'un client dans un modèle imbriqué peut croître sensiblement sans véritable augmentation du temps d'attente (dans la file principale), ce qui implique qu'il peut commencer à croître dès les premiers niveaux de charge. Comme l'illustre la Figure 3.5, à faible niveau de charge les augmentations du temps de séjour moyen sont essentiellement dues à une élongation du temps de service, et à mesure que la charge grandit, cette augmentation résulte de l'attente croissante des clients dans la file.

Le fonctionnement des modèles imbriqués vise à représenter de façon simple le phénomène d'élongation du temps de service que l'on retrouve dans un certain nombre de systèmes informatiques. Pour un système donné l'*élongation de son temps de service* résulte généralement de l'indisponibilité temporaire d'une ressource nécessaire à l'exécution de son service ; ainsi, l'élongation tend habituellement à s'accroître lorsque le niveau de la charge du système augmente. Nous présentons trois exemples de systèmes pour lesquels le temps de service s'accroît avec la charge. Premièrement, le temps que met un routeur pour servir un paquet peut augmenter à mesure que le taux d'arrivée des paquets à router augmente car le routeur consacre alors une partie croissante de son temps (i.e. de ses ressources) à ordonner les arrivées de paquets dans sa file.

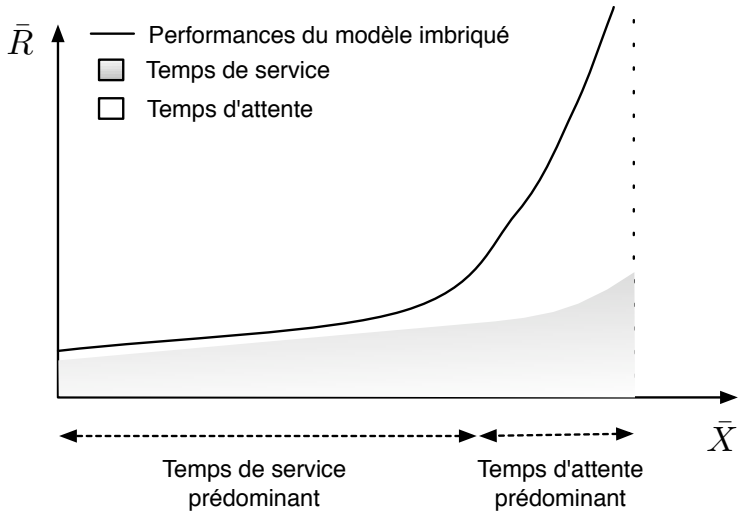


FIG. 3.5 – Répartition du temps de service et du temps d'attente pour un modèle imbriqué

Deuxièmement, le temps de service d'un poste faisant partie d'un système distribué comprend généralement le traitement par une ressource commune à tous les postes. Ainsi, il est probable que lorsque la charge de travail engendrée par les postes augmente, des contentions d'accès se produisent sur la ressource partagée entraînant la formation de files d'attente et entraînant donc au niveau des postes une élongation du temps de service. Troisièmement, il est connu que le temps de service des clients les moins prioritaires dans des systèmes à temps partagés multiclasses du fait des clients plus prioritaires tend à s'allonger lorsque la charge du système augmente [25].

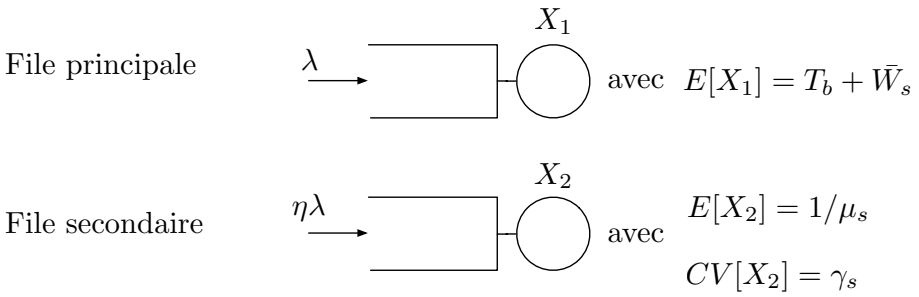


FIG. 3.6 – Paramètres structurels et paramètre de charge pour un modèle imbriqué

Pour les exemples que nous présentons dans le Chapitre 4, l'ensemble des briques considéré par notre méthode HLM ne comporte qu'un seul modèle imbriqué. Il s'agit de la forme la plus simple du modèle imbriqué. Les files principale et secondaire sont toutes

les deux représentées par des files M/G/1 à capacité d'accueil illimitée. Le temps de service pour la file secondaire est égal en moyenne à $1/\mu_s$ et son coefficient de variation est défini par γ_s. D'autre part, T_b, le temps de service de base de la file principale est une constante. Par conséquent dans sa forme la plus simple, le modèle imbriqué a quatre paramètres structurels : T_b, η, μ_s et γ_s et un seul paramètre de charge : λ. La Figure 3.6 illustre le rôle de ces paramètres pour cette file imbriquée. Sur cette figure, X_1 et X_2 représentent respectivement les lois de service de la file principale et secondaire.

Le calcul des paramètres de performances moyens de la file principale en régime permanent nécessite d'évaluer (a) son temps de service moyen S_p et (b) le coefficient de variation de son temps de service γ_p. Ensuite, en utilisant les formules connues pour une file M/G/1 [56, 2, 5], on obtient les paramètres de performances moyens de cette brique :

$$\begin{aligned} \bar{R} &= S_p + \frac{\lambda S_p^2(1+\gamma_p^2)}{2(1-\lambda S_p)} \\ \bar{X} &= \lambda \\ \bar{Q} &= \bar{R}\bar{X} \end{aligned}$$

A présent, on montre comment calculer S_p et γ_p. Rappelons que W_s désigne le temps d'attente dans la file secondaire et $\rho_s = \lambda_s/\mu_s$.
Pour S_p, on a :

$$\begin{aligned} S_p &= T_b + \bar{W}_s \\ &= T_b + \frac{\lambda_s(1+\gamma_s^2)}{2\mu_s^2(1-\rho_s)} \end{aligned} \tag{3.1}$$

Pour (3.1), on utilise simplement l'expression connue du temps d'attente moyen dans la file M/G/1 secondaire $\bar{W}_s$ [56, 2, 5].

Pour γ_p, on a par définition :

$$\begin{aligned}
\gamma_p &= \frac{\sqrt{Var(T_b + W_s)}}{T_b + \bar{W}_s} \\
&= \frac{\sqrt{Var(W_s)}}{T_b + \bar{W}_s} \qquad (3.2) \\
&= \frac{\sqrt{E[{W_s}^2] - {\bar{W}_s}^2}}{T_b + \bar{W}_s} \\
&= \frac{\sqrt{{\bar{W}_s}^2 + \lambda_s \frac{E[S_p^3]}{3(1-\rho_s)}}}{T_b + \bar{W}_s} \qquad (3.3)
\end{aligned}$$

Pour (3.2), d'une part T_b et W_s étant indépendantes, la variance de leur somme est égale à la somme de leurs variances, et d'autre part la variance d'une constante, i.e. T_b est nulle. Pour (3.3), on a simplement utilisé l'expression analytique connue [2] du moment d'ordre 2 du temps d'attente dans une file M/G/1, i.e. $E[{W_s}^2] = 2{\bar{W}_s}^2 + \lambda_s \frac{E[S_p^3]}{3(1-\rho_s)}$ où $E[S_p^3]$ représente le moment d'ordre 3 du temps de service de la file secondaire. Les deux premiers moments du temps de service sont directement accessibles à partir de μ_s et γ_s mais le moment d'ordre 3 lui est inconnu[3]. On détermine $E[S_p^3]$ en supposant que la loi de service suit une distribution hyperexponentielle, hypoexponentielle (i.e. Erlang généralisée) ou exponentielle, selon que γ_s est supérieur, inférieur ou égal à 1, ajustée sur les deux premiers moments connus du temps de service.

Il existe quelques propriétés remarquables pour les paramètres de performances de cette file.

1. $\bar{X} < \frac{\mu_s}{\eta}$: le débit moyen en sortie de la file principale ne peut pas être supérieur à $\frac{\mu_s}{\eta}$ car sinon la file secondaire est instable ;
2. $\bar{X} < \frac{1}{T_b + \bar{W}_s}$: pour un niveau de charge donné λ, le débit moyen en sortie de la file principale ne peut pas être supérieur à son taux de service (pour cette charge-là) ;
3. $T_b \leq \bar{R}$: le temps de séjour d'un client dans la file principale comprend au moins la composante constante de base.

[3]Il aurait été possible de faire du moment d'ordre 3 du temps de service de la file secondaire un paramètre structurel supplémentaire de la brique.

La file M/M/Cv

Nous avons également défini un autre type de brique non-standard, que l'on désigne comme la file *M/M/Cv*, qui représente une file multiserveur pour laquelle le nombre de serveurs change en fonction de l'utilisation de la ressource. En pratique, ce modèle se comporte comme une file M/M/C excepté que le nombre de serveurs passe de C_1 à C_2 si le taux des arrivées dans la file dépasse un certain seuil θ. Il s'agit donc d'un modèle ouvert formé d'une file FIFO et de plusieurs serveurs. La capacité d'accueil de la file est illimitée. Les clients sont émis depuis une seule source et leurs arrivées dans la file suivent un processus de Poisson de taux λ. Le temps de service des clients dans chacun des serveurs est distribué selon une loi exponentielle de moyenne $1/\mu$. La particularité de ce modèle est que le nombre de serveurs de la file vaut C_1 si $\lambda \leq \theta$ et C_2 sinon avec $C_2 \geq C_1$.

Nous avons ajouté cette brique à notre ensemble de briques avec l'idée qu'elle pourrait permettre de représenter, avec un certain niveau d'abstraction, le comportement des systèmes qui octroient des ressources supplémentaires lorsque la demande augmente.

Le calcul des paramètres de performances moyens de cette file est identique à celui d'une file standard M/M/C à ceci près que la valeur de C dépend du signe de $(\lambda - \theta)$. En définissant $\rho = \lambda/\mu$, on obtient :

$$\begin{aligned} \bar{X} &= \lambda \\ \bar{R} &= \frac{1}{\mu} + \sum_{n=C}^{\infty} p(n) \frac{n+1-C}{C\mu} \\ \bar{Q} &= \rho + p(0) \frac{\rho^{C+1}}{(C-1)!(C-\rho)^2} \end{aligned}$$

avec $C = C_1$ si $\lambda \leq \theta$ et $C = C_2$ sinon.

La file M/M/Cv possède quatre paramètres structurels : C_1, C_2, θ et μ et un paramètre de charge : λ. Ainsi, lorsqu'on compare un calibrage de la file M/M/Cv à un jeu de mesures, une valeur différente de λ doit être considérée pour chacun des points de mesure (tandis que les valeurs de C_1, C_2, θ et μ elles restent fixes).

Il existe quelques propriétés remarquables pour les paramètres de performances de cette file.

1. $\bar{X} < C_2\mu$: le débit moyen en sortie de la file ne peut pas être supérieur à la capacité maximale de traitement de la file ;
2. $1/\mu \leq \bar{R}$: le temps de séjour d'un client comprend au moins son temps de service.

3.2.5 Le calibrage de chaque brique

Le calibrage de chacune des briques constitue la deuxième étape de notre méthode HLM. Cette étape consiste essentiellement à déterminer pour chaque brique la combinaison de ses paramètres qui lui permet de reproduire « au mieux » le comportement décrit par les points de mesure.

La méthode adoptée

Le calibrage de chaque brique est réalisé grâce à la méthode de calibrage que nous avons décrite dans le Chapitre 2 de ce rapport. Cette méthode présente plusieurs avantages qui en font un choix particulièrement bien adapté ici. Premièrement, son algorithme de recherche s'avère être efficace lorsque le modèle concerné possède moins d'une dizaine de paramètres, ce qui est le cas de toutes les briques présentées dans ce chapitre. Deuxièmement, son fonctionnement ne requiert aucun préalable spécifique à la brique considérée puisqu'aucune propriété particulière du modèle à calibrer n'est supposée si ce n'est de pouvoir évaluer ses paramètres de performances. Elle peut donc être appliquée à toutes les briques présentées précédemment. Troisièmement, elle a été pensée pour fonctionner pour une grande variété de jeux mesures ce qui permet d'assurer le fonctionnement de la méthode HLM pour des jeux de mesures divers et variés. Enfin, cette méthode de calibrage permet de prendre en compte simplement et efficacement des contraintes garantissant la cohérence entre le calibrage recherché du modèle et les mesures. A notre connaissance (cf. Chapitre 2), aucune autre méthode de calibrage ne réunit à la fois tous ces avantages.

La mise en oeuvre

Nous renvoyons le lecteur qui désire connaître plus précisément la mise en oeuvre de cette méthode de calibrage au Chapitre 2 de ce rapport qui lui est entièrement consacré. Toutefois, nous présentons dans cette section les deux points essentiels permettant d'utiliser cette méthode de calibrage pour la méthode HLM : mettre en place les contraintes et décider de la fonction objectif δ.

Les contraintes. Les contraintes visent principalement à assurer que le calibrage recherché de la brique est cohérent avec les points de mesure. Elles garantissent que les paramètres de performances mesurés correspondent à des niveaux atteignables par le modèle. La plupart de ces contraintes résultent directement des propriétés remarquables que nous avons présentées pour chacune des briques. Par exemple, pour une

brique M/M/C/K munie d'un d'offset, les contraintes peuvent permettre de garantir que tous les temps de séjour moyens mesurés correspondent à des niveaux pouvant être engendrés par le modèle (soumis à certains niveaux de charge), que le nombre moyen de clients mesurés ne dépasse pas la capacité d'accueil de la file ou encore que le débit maximal mesuré en sortie du système est inférieur à la capacité maximale de traitement du modèle. Notons enfin que certaines contraintes peuvent être le résultat d'une connaissance partielle du système ; l'utilisateur peut par exemple souhaiter que la valeur de certains paramètres du modèle soit comprise dans un intervalle donné.

Nous expliquons à présent comment ces propriétés remarquables des briques sont converties en contraintes de la méthode de calibrage. Rappelons d'abord que la méthode de calibrage décrite dans le Chapitre 2 fonctionne par améliorations successives d'un calibrage donné. Lorsqu'un des calibrages considérés par la méthode est tel qu'au moins un des points de mesure est en contradiction avec une des propriétés remarquables de cette brique, le calibrage est qualifié d'« orange » ou de « rouge » (cf. Chapitre 2, Section 2.4.2, page 33) selon qu'il se trouve ou non dans le domaine de définition de la fonction objectif. S'il est orange, et c'est une des propriétés originale de l'algorithme DFO impliqué dans la méthode, le calibrage concerné bien que violant au moins une des contraintes peut être utilisé pour guider la recherche vers le meilleur calibrage (cf. Chapitre 2, Section 2.5.2, page 45). S'il est rouge, le calibrage est simplement ignoré par l'algorithme de recherche DFO.

L'intégration de ces contraintes (non-linéaires pour certaines) à la recherche du calibrage s'effectue donc très simplement et leur présence n'entrave pas le bon fonctionnement de notre méthode de calibrage. Cela constitue un avantage certain de notre méthode de calibrage sur la plupart des autres approches existantes pour lesquelles l'existence de contraintes non-linéaires constituent généralement un obstacle (cf. Chapitre 2, Section 2.5.1, page 42).

La fonction objectif. Après avoir fixé les contraintes, il reste à décider de la fonction objectif δ. Comme nous l'avons expliqué dans le Chapitre 2, cette décision se fait en deux étapes : le choix des points couplés et le choix de la métrique.

La métrique. Concernant le choix de la métrique, nous recommandons la métrique multi-critères (formule (2.3), page 38), pour les raisons décrites dans le Chapitre 2.

Les points couplés. Les points couplés définissent les états du modèle que l'on associe aux points de mesure pour le calcul de la métrique. En effet, puisque qu'on suppose que chaque point de mesure correspond à un niveau de charge différent du système, le calibrage recherché de chaque brique doit également être considéré à plusieurs niveaux de charge, c'est-à-dire pour des valeurs différentes de ses paramètres de charge. Il s'agit donc d'inférer les valeurs des paramètres de charge de la brique considérée pour chacun des points de mesure. Nous optons, comme nous l'avons préconisé dans le Chapitre 2 (Section 2.4.3, page 35), pour une approche basée sur un paramètre-critère : le point couplé et le point de mesure ont la même valeur pour ce paramètre-critère. Par exemple, considérons le calibrage de la brique imbriquée sur le premier exemple de jeu de mesures présenté dans le Chapitre 1 (Table 1.1, page 21). Si le débit moyen $\bar{X}$ est choisi pour paramètre-critère, la sélection des points couplés suppose d'inférer les valeurs du paramètre de charge λ du modèle tels que son débit moyen soit égal à celui de chaque point de mesure. Cette inférence s'effectue simplement avec une technique comme la bisection car $\bar{X}$ a une croissance monotone avec le niveau de la charge pour la file imbriquée comme pour d'ailleurs la plupart des modèles de type file d'attente. Notons que dans le cas où le niveau de charge du système associé à chaque point de mesure est connu, ou plus précisément lorsque la variation de la charge d'un point de mesure à un autre est connue, aucune inférence n'est requise pour décider des points couplés. C'est par exemple le cas du deuxième exemple dans le Chapitre 1 (Table 1.2, page 22) pour lequel est indiqué, en plus du temps moyen de séjour d'un client, le nombre de sources qui étaient connectées au serveur central. Enfin, si certains paramètres de charge de la brique ne sont pas associés aux variations de la charge, c'est-à-dire qu'ils restent constants pour tous les points de mesures, alors ces paramètres se joignent à l'ensemble des paramètres indéterminés de la brique qu'il faut calibrer.

3.2.6 La sélection du modèle lauréat

Le modèle lauréat de notre méthode correspond simplement à la brique calibrée qui reproduit au mieux le jeu de mesures. Pour le désigner, il suffit de classer les briques en fonction de leur distance aux mesures ; le premier de cette liste correspond au modèle lauréat. Pour déterminer cette distance, on peut simplement reprendre la fonction objectif qui a été utilisée pour rechercher le meilleur calibrage de chaque brique ou bien adopter n'importe quel autre critère de comparaison pourvu que le calibrage trouvé de chacune des briques soit (ré-)évalué selon ce même critère.

En pratique, lorsque plusieurs briques calibrées présentent des niveaux de qualité équivalents, ce qui signifie qu'ils reproduisent de manière comparable le jeu de mesures,

nous privilégions comme modèle lauréat le modèle ayant le plus petit nombre de paramètres (comme préconisé par le principe de parcimonie [21]). Cette préférence peut être automatiquement prise en compte par la fonction objectif choisie en faisant intervenir dans le critère de comparaison retenu un facteur pénalisant pour les briques les plus compliquées.

3.3 Des modèles simples pour des systèmes au fonctionnement compliqué

Comme nous l'avons dit, le principe fondateur de la méthode HLM consiste à supposer l'existence d'un modèle relativement simple pour reproduire les performances d'entrée/sortie d'un système dont le fonctionnement interne peut être très compliqué. C'est le pari qui est fait par notre méthode. Nous présentons deux exemples de systèmes d'attente pour lesquels l'existence d'un modèle relativement simple peut paraître a priori très improbable.

Le premier exemple que nous considérons est une file M/G/1 avec une discipline de service à priorité préemptive (« preemptive-resume priority »). On suppose qu'il existe trois niveaux de priorités ; le niveau 1 est le plus prioritaire et le niveau 3 le moins prioritaire. On désigne par λ_i le taux des arrivées de priorité i, et par $1/\mu_i$ et γ_i respectivement la moyenne et le coefficient de variation du temps de service pour le niveau i. Pour commencer, on s'intéresse au temps moyen de réponse pour le niveau à la priorité la plus basse. On considère ces temps moyens de réponse pour plusieurs valeurs de λ_3 en supposant que la charge provenant des autres niveaux de priorité est constante. La solution connue pour une file M/G/1 à priorité [2] permet de calculer les paramètres de performances pour l'exemple représenté sur la Figure 3.7. Même si l'on dispose d'une expression analytique pour ses paramètres de performances, il n'est pas évident que le temps de réponse pour le niveau de priorité considéré corresponde en fait à celui d'une simple file M/G/1 avec un coefficient de variation différent (supérieur) munie d'un offset comme l'illustre la Figure 3.7. Il peut également être intéressant d'observer que pour les valeurs des paramètres utilisées ici, une simple file M/M/1 (comme le proposent certaines approximations), même avec un offset, ne permet pas de représenter correctement le comportement des niveaux de priorité les plus bas (cf. [51]). Précisons que la recherche des paramètres adéquats pour ces deux files a été faite par la méthode de calibrage automatique décrite dans le Chapitre 2.

Le deuxième exemple concerne un réseau tandem ouvert de files à capacité d'accueil finie. Les blocages sur ce réseau sont ceux de type communication i.e. le service à un

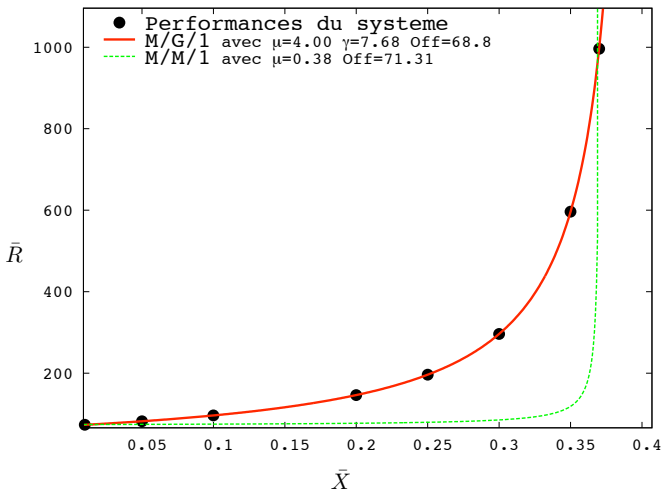


FIG. 3.7 – Comportement de la classe la moins prioritaire d'une file M/G/1 à priorité avec $(\mu_1 = 0.1, \gamma_1 = 2)$, $(\mu_2 = 0.5, \gamma_2 = 2)$ et $(\mu_3 = 1, \gamma_3 = 2)$ et en supposant la charge des classes plus prioritaires constante

noeud ne commence pas tant qu'une place n'est pas disponible pour le recevoir dans le buffer du noeud suivant. Ce réseau est représenté sur la Figure 3.8. Nous supposons que les arrivées vers ce système proviennent d'une source de Poisson de taux λ, que les temps de service à chaque noeud sont distribués exponentiellement, et nous désignons par μ_i le taux de service du noeud i. Nous désignons par B_i la taille du buffer du noeud i. Nous choisissons les valeurs des B_i, des μ_i et de λ de façon à avoir une quantité significative de blocage sur ce réseau. Dans ces conditions, on peut supposer que les performances de ce réseau tandem s'écartent sensiblement de celles d'un système d'attente Jacksonien [46]. On utilise l'approximation développée par Brandwajn et Jow [19] pour résoudre ce réseau de files ouvert à trois neuds. La Figure 3.9 montre le temps de séjour total dans un réseau de ce type pour plusieurs valeurs de débit atteint en sortie. Il est intéressant de remarquer, comme l'illustre la Figure 3.9, qu'une simple file M/G/1 munie d'un offset parvient à reproduire finement le comportement du temps de séjour total moyen d'un tel système à des niveaux différents de débits moyens. Notons cependant qu'une file à capacité d'accueil illimitée est naturellement inapte à modéliser le taux de rejet subis par les requêtes au niveau du premier noeud.

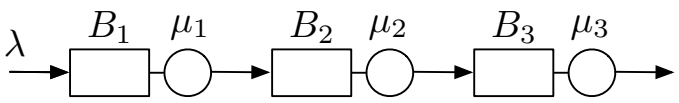


FIG. 3.8 – Réseau tandem ouvert avec des files à capacité d'accueil finie

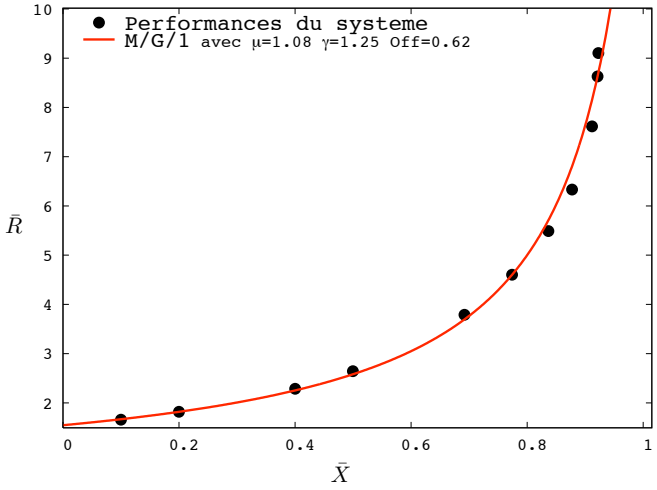


FIG. 3.9 – Comportement d'un réseau tandem ouvert avec $(B_1 = 5, \mu_1 = 5)$, $(B_2 = 3, \mu_2 = 3)$ et $(B_3 = 2, \mu_3 = 1)$

A présent, on revient au premier exemple, i.e. la file M/G/1 à trois niveaux de priorité. On considère à nouveau les performances de la classe de priorité la plus basse mais cette fois-ci, d'un point de mesure à un autre, on augmente à proportion égale le taux d'arrivée pour tous les niveaux de priorité. La Figure 3.10 montre le temps de séjour moyen des clients de niveau de priorité 3. Ici, aucune file M/G/1 ne peut reproduire les temps de réponses observés car les délais qui impactent la classe la moins prioritaire s'allongent à mesure que son taux d'arrivée λ_3 augmente : autrement dit, il se produit une élongation du temps de service de la classe la moins prioritaire. Toutefois dans cet exemple, on observe que la file imbriquée permet de reproduire finement les performances du niveau de priorité le plus bas.

La Figure 3.11 représente le temps de séjour moyen pour le noeud du milieu dans le réseau tandem de trois noeuds que nous avons présenté ci-dessus. Nous observons qu'aucune file simple M/M/C ou M/G/1 ne parvient à reproduire correctement le comportement de ce noeud. On observe cependant que la file imbriquée nous permet de reproduire fidèlement le comportement du noeud du mileu de ce réseau tandem à trois noeuds.

Enfin, notons que, même si nous avons montré qu'une file M/G/1 munie d'un offset permet de représenter avec précision le comportement du niveau de priorité le plus bas d'une file à priorité M/G/1 (cf. Figure 3.7), dans notre approche on n'est pas capable de dériver directement les paramètres de cette file à partir des paramètres du « système source » comme cela pourrait être le cas pour une modélisation constructive.

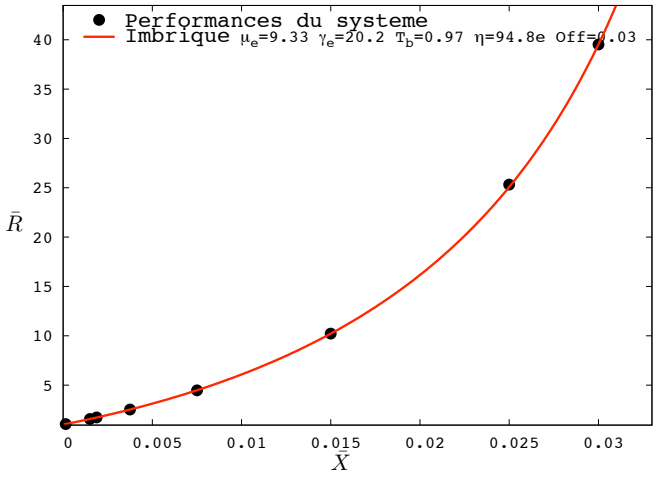


FIG. 3.10 – Comportement de la classe la moins prioritaire d'une file M/G/1 à priorité avec $(\mu_1 = 0.1, \gamma_1 = 2)$, $(\mu_2 = 0.5, \gamma_2 = 2)$ et $(\mu_3 = 1, \gamma_3 = 2)$ et en supposant des variations de la charge des classes plus prioritaires

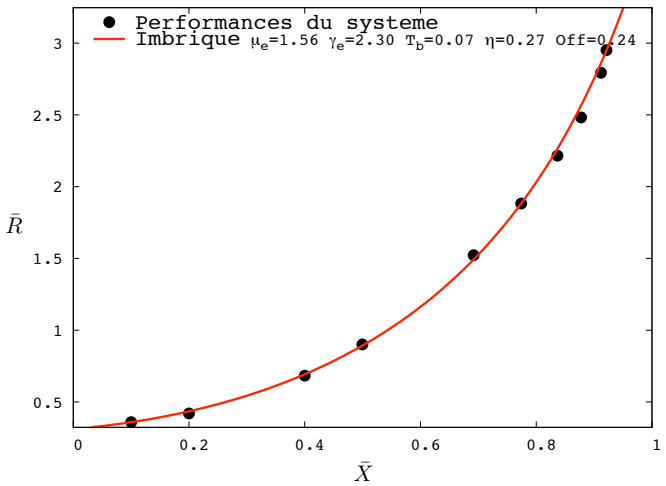


FIG. 3.11 – Comportement du noeud du milieu d'un réseau tandem ouvert avec $(B_1 = 5, \mu_1 = 5)$, $(B_2 = 3, \mu_2 = 3)$ et $(B_3 = 2, \mu_3 = 1)$

Ceci restreint la capacité prédictive de notre approche au cas où le débit de sortie du système source change (si un des paramètres structurels du système source change, notre approche ne permet pas de prévoir son comportement). Toutefois, le simple fait qu'une file M/G/1 (pour cet exemple) constitue un bon candidat pour modéliser le comportement du système et pas une file M/M/1, peut servir pour la recherche d'un modèle constructif simple pour ce système. A ce sujet, notons que plusieurs auteurs [60, 59] ont considéré l'utilisation d'une file M/G/1 comme point de départ pour modéliser des réseaux de files d'attente.

Plus largement, notons qu'une façon d'approcher le comportement des systèmes multiclasses à priorité consiste à reproduire les préemptions causées par les classes plus prioritaires sur les classes moins prioritaires dans le modèle en mettant de temps en temps le serveur du système en pause. Ainsi, Nain propose d'approcher le comportement de chacune des classes d'une file multiclasse à priorité préemptive avec un seul serveur par une file G/G/1 (monoclasse) à interruption de service [75]. Les interruptions de service du serveur représentent dans son modèle les effets des classes plus prioritaires sur celles moins prioritaires. Deux remarques peuvent être faites à ce sujet. D'une part, les résultats de cet article corroborent l'idée fondatrice de cette thèse en fournissant un exemple supplémentaire d'un modèle relativement simple permettant de reproduire le comportement d'un système au fonctionnement complexe. Notons d'ailleurs qu'il est envisageable d'ajouter ce modèle à interruption de service comme une brique de la méthode HLM. D'autre part, cette technique [75] se démarque de la méthode HLM car ici l'auteur est parvenu à dériver des formules simples afin de déterminer les paramètres de la file approchante à partir des paramètres du système source considéré. A l'inverse, dans l'approche HLM, les paramètres de la « de remplacement » sont découverts non pas en fonction des paramètres structurels du système source (puisqu'on les considère comme inconnus) mais en fonction de ses performances d'E/S. D'une façon similaire, la technique du « Shadow Server » [51] (également appelée approximation ROA « Reduced Occupancy Approximation ») peut être rapprochée de la méthode HLM ; cette technique permet d'approcher le comportement de systèmes multiclasses à priorité en ralentissant cette fois-ci la vitesse des serveurs afin de reproduire les effets des classes les plus prioritaires.

Par ailleurs, nous avons expérimenté avec succès notre méthode HLM sur de nombreux jeux de mesures réels que nous présentons dans le chapitre suivant. Bien que nous ayons rencontré peu de jeux de mesures réels pour lesquels la méthode n'a pas fonctionné, il est clair qu'il existe des contre-exemples pour lesquels (1) l'ensemble des briques ne permet pas de trouver un modèle proche des mesures, (2) les prévisions déli-

vrées par le modèle de notre méthode sont inexactes. Cependant, nous pensons que ce qui importe vraiment, c'est la proportion de cas pour lesquels la méthode fonctionne et, d'après notre expérience, ce taux semble (peut être de façon surprenante) suffisamment élevé pour rendre la méthode HLM attractive.

Les systèmes pour lesquels a été appliquée la méthode HLM comprennent des contrôleurs disques, des serveurs web, des serveurs centraux, des processeurs, des réseaux filaires (de type Ethernet), des réseaux sans-fil (de type Wifi ou large bande) ainsi que des chemins UDP sur des réseaux maillés sans-fil. En se basant uniquement sur un jeu de mesures de chacun de ces systèmes, la méthode HLM est parvenue à trouver un modèle relativement simple de type file d'attente qui modélise correctement le système. Nous présentons ces exemples ainsi que les applications possibles du modèle produit par la méthode HLM dans le chapitre suivant.

3.4 Conclusion : atouts et limites de cette approche

Dans ce chapitre, nous avons présenté une méthode de modélisation entièrement automatique qui repose uniquement sur des mesures. Cette méthode HLM recherche un modèle relativement simple de type file d'attente pour représenter le comportement d'un système décrit par un jeu de mesures. Le principe général de la méthode HLM consiste à chercher parmi un ensemble de modèles génériques de type file d'attente, i.e. les briques, celui dont le calibrage ad hoc permet de reproduire au mieux le jeu de mesures considéré. L'ensemble des modèles génériques qui permet de représenter un spectre assez large de comportements observés sur les systèmes informatiques ainsi que la méthode de calibrage automatique constituent les deux ressorts principaux de cette méthode HLM.

Le fonctionnement de cette méthode implique un certain nombre de limites. Certaines de ces limites sont liées au jeu de mesures tandis que les autres résultent de l'automaticité de la méthode. Premièrement, la nécessité des mesures suppose que le système considéré par la méthode existe et soit instrumenté (à moins que les performances du système ne soient engendrées par une simulation du système). Deuxièmement, le jeu de mesures doit comprendre les aspects clés du comportement du système, du moins ceux qui se situent dans la plage de fonctionnement à laquelle s'intéresse le modèle. Par exemple, si le temps de réponse d'un système exhibe un point d'inflexion et que celui-ci n'apparaît pas dans le jeu de mesures, il y a alors très peu de chance que le modèle produit par notre méthode HLM reproduise un tel comportement. Troisièmement, la méthode ne garantit pas qu'un modèle puisse être trouvé pour chaque système

puisqu'elle recherche le modèle parmi un ensemble fini de briques. Il se peut qu'aucune des briques considérées ne soit adaptée ou bien que le comportement du système ne soit tout simplement pas représentable par un modèle relativement simple. Enfin, les modèles produits par la méthode HLM ne sont pas aussi « réalistes » que ceux produits par une modélisation constructive. Il peut par exemple être difficile d'associer un sens précis aux valeurs de leurs paramètres.

Ceci étant dit, la méthode HLM présente des atouts. Premièrement, son exploitation ne suppose aucune connaissance intrusive sur le système, ni aucune compétence particulière en modélisation. En effet, la méthode s'appuie uniquement sur des mesures d'entrée/sortie du système pour rechercher un modèle ce qui permet de rendre son utilisation accessible au plus grand nombre. Deuxièmement, le fonctionnement général de la méthode et en particulier sa méthode de calibrage permet de rendre notre méthode entièrement automatique. Par conséquent, la méthode HLM peut être embarquée dans un outil de modélisation automatique, rapide et simple à utiliser. Troisièmement, comme nous le montrons dans le chapitre suivant de ce rapport, les modèles produits par notre méthode peuvent avoir diverses fonctions. Ainsi, le modèle trouvé par la méthode peut être utile pour prévoir le comportement du système à des niveaux de charge non mesurés par les points de mesure, être utile dans la modélisation d'un système plus vaste dont fait partie le système étudié ou encore être utile pour guider une modélisation constructive. Enfin, le fonctionnement de la méthode en fait naturellement une méthode modulaire et extensible. De nouvelles briques peuvent être ajoutées très simplement à la méthode puisqu'aucune des étapes de la méthode ne requiert de préalables ou d'hypothèse particulière sur le modèle. Cette modularité permet notamment d'étendre l'ensemble des briques lorsqu'un jeu de mesures présente un comportement qualitatif différent de ceux des briques actuelles ou bien encore d'ajouter une brique ad hoc résultant d'une connaissance supplémentaire du système.

Notons qu'une des hypothèses essentielles concernant le système considéré par la méthode HLM (ainsi que par la méthode de calibrage proposée) consiste à supposer qu'à une charge donnée correspond un niveau de performances donné. Toutefois, il est plus que probable qu'en réalité, en re-soumettant une même charge au système, les performances mesurées ne soient pas exactement les mêmes. Cet écart peut s'expliquer (1) par l'incertitude intrinsèque aux mesures ou (2) par un changement des conditions de fonctionnement du système. Si cet écart est dû à un changement de fonctionnement du système, c'est que le système ou son environnement ont changé d'un point de mesure à un autre. Il est alors probable que deux modèles distincts soient nécessaires pour reproduire ces deux points de mesure. Par conséquent, dans sa forme actuelle, la méthode

HLM risque d'échouer car elle présuppose l'existence d'un seul modèle (simple) pour reproduire tous les points de mesure[4]. La méthode de calibrage automatique risque elle aussi d'échouer car elle présuppose également un modèle unique pour tous les points de mesure. En revanche, si cet écart résulte de l'imprécision ou de la variabilité naturelles des mesures, la méthode HLM et la méthode de calibrage peuvent fonctionner malgré ces possibles « doublons ». En effet, dans le cas où le jeu de mesures étudié comporte deux points de mesure associés à un même niveau de charge, la méthode HLM optera naturellement pour la brique calibrée qui passe au plus près de ces deux points, c'est-à-dire approximativement entre les deux. La méthode de calibrage « moyenne » donc intrinsèquement ces doublons. A ce sujet l'exemple présenté à la Figure 4.11(b) (cf. Chapitre 4, page 101) illustre ce comportement. Notons que si l'on dispose de plusieurs points semblant correspondre à des niveaux de charges identiques, mais qu'on accorde plus de crédit à certains de ces points, on peut décider de moduler l'impact de ces points dans la recherche du calibrage par l'intermédiaire des coefficients pondérateurs de la fonction objectif δ (cf. Section 2.4.3, page 37).

Au final la méthode HLM présente des avantages qui nous paraissent inhabituels pour une méthode de modélisation mais en contre-partie, elle a également un certain nombre de limites inhérentes à son fonctionnement. En pratique, la majorité des expérimentations que nous en avons faite sur des jeux de mesures réels suggère que nous avons atteint nos objectifs initiaux : développer une méthode de modélisation et de calibrage automatique.

[4]Notons qu'une façon de tenir compte de ces écarts dans la méthode HLM pourrait être d'intégrer la possibilité d'une incertitude sur l'environnement et les conditions de fonctionnement du système.

Chapitre 4

Applications de la méthode HLM

Sommaire

4.1 Introduction

Ce chapitre présente des exemples d'utilisation de la méthode HLM. Les jeux de mesures considérés dans ce chapitre sont issus de systèmes informatiques et de réseaux de communication divers et variés. Pour chaque exemple, nous avons présenté en plus du modèle lauréat proposé par notre méthode, des exemples d'applications possibles du modèle.

Afin d'évaluer le pouvoir prédictif des modèles proposés, nous avons recours aux *points cachés*. Un point caché désigne simplement un point que l'on retire volontairement de son jeu de mesures avant d'effectuer la recherche du modèle lauréat. Une fois trouvé, on évalue la capacité prédictive de ce modèle en comparant ses performances à celles du point de mesure caché.

Enfin comme nous l'avons expliqué dans le Chapitre 2, puisque que le calibrage de chaque brique s'effectue par une méthode numérique, les modèles délivrés par la méthode HLM peuvent avoir des valeurs réelles pour des paramètres habituellement entiers.

4.2 Un réseau sans fil à large bande

Le premier exemple concerne un réseau sans fil à haut-débit pour lequel Quintero et al. ont obtenu un jeu de mesures [86]. Chaque point de mesure associe le débit moyen de paquets transmis par le réseau $\bar{X}$ au temps de séjour moyen des paquets dans le réseau $\bar{R}$. Les points de mesure ont été obtenus pour plusieurs niveaux de charge (non mesurés) et constituent ensemble le jeu de mesures illustré par la Figure 4.1. Comme nous l'avons dit précédemment, nous cachons un certain nombre de points de ce jeu de mesures avant de lancer la recherche du modèle lauréat. La disposition des points de mesure (cf. Figure 4.1) suggère que la zone de fonctionnement du réseau la plus difficilement prévisible est probablement celle située aux environs du plus haut niveau de charge. C'est pourquoi nous décidons de cacher justement du jeu de mesures le point correspondant au plus grand débit de façon à évaluer les capacités prédictives du modèle proposé par la méthode HLM dans ce qui nous paraissent être les conditions les plus difficiles.

La Figure 4.1 montre qu'une simple file M/G/1, avec des paramètres adéquats (i.e. $\mu = 20.03$, $\gamma = 5.5$ et $\mathit{Off} = 0.44$) parvient à reproduire finement les performances de ce réseau sans fil. Le rôle du paramètre offset désigné par Off a été décrit dans le Chapitre 3 (cf. Section 3.2.3, page 69). Nous avons également représenté sur cette figure le résultat

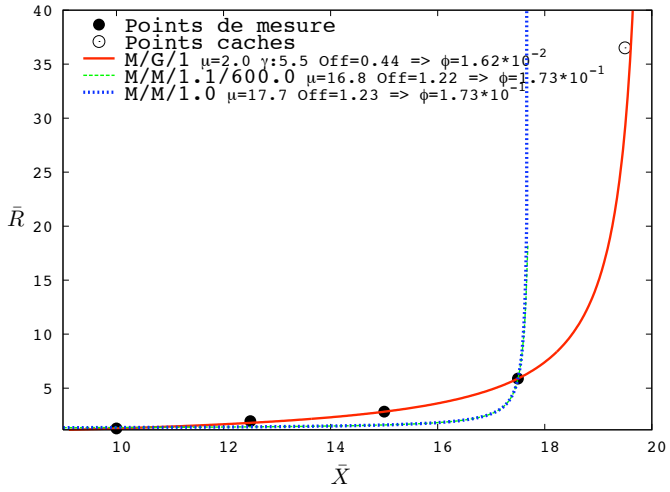


FIG. 4.1 – Un réseau sans fil large bande

du meilleur calibrage possible d'une file M/M/C et M/M/C/K (les courbes associées à ces deux modèles sont difficiles à différencier car elles se superposent). Nous observons qu'aucune de ces deux files ne parvient à reproduire correctement le comportement du système.

Par ailleurs, la file lauréate M/G/1 permet également de prédire avec une précision satisfaisante les performances du réseau au niveau du point de mesure caché. En comparant le temps de séjour moyen atteint par le réseau à celui prédit par la file M/G/1 pour le niveau de débit du dernier point de mesure, l'erreur relative est de 15%, tandis qu'elle est inférieure à 1% si l'on compare le débit moyen mesuré sur le réseau et celui évalué par le modèle pour le même niveau de temps de séjour. Etant donnée la pente raide qu'ont les temps de séjour dans le voisinage du point caché, nous pensons que la précision atteinte par le modèle lauréat est plus que raisonnable. Au passage, on note que les prédictions délivrées par les briques calibrées M/M/C et M/M/C/K sont elles très mauvaises. Enfin, en déplaçant le point caché sur un autre point de mesure, le modèle proposé par la méthode HLM reste une file M/G/1 avec des paramètres très proches de ceux indiqués ci-dessus et les erreurs relatives de ses prédictions sont toutes inférieures à 5%.

4.3 Des contrôleurs disques

A présent, on considère deux jeux de mesures qui ont été obtenus sur des contrôleurs disques opérant pour des serveurs centraux. Chaque point de mesure associe un temps

de réponse moyen pour une requête d'E/S $\bar{R}$ à un débit moyen atteint d'E/S (en nombre de requêtes par unité de temps) $\bar{X}$.

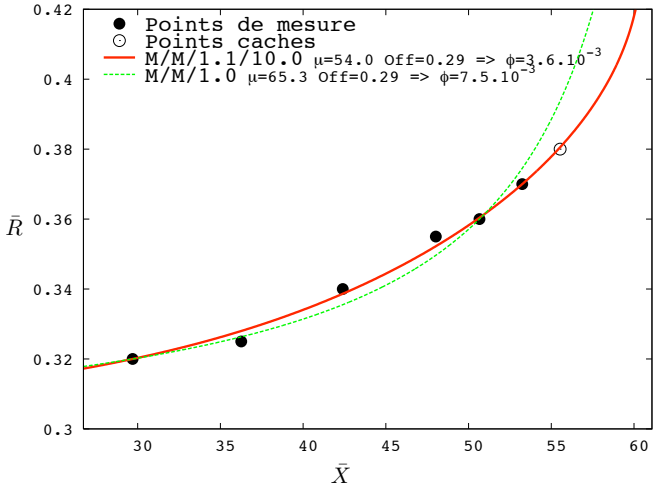


FIG. 4.2 – Un contrôleur disque - Jeu 1

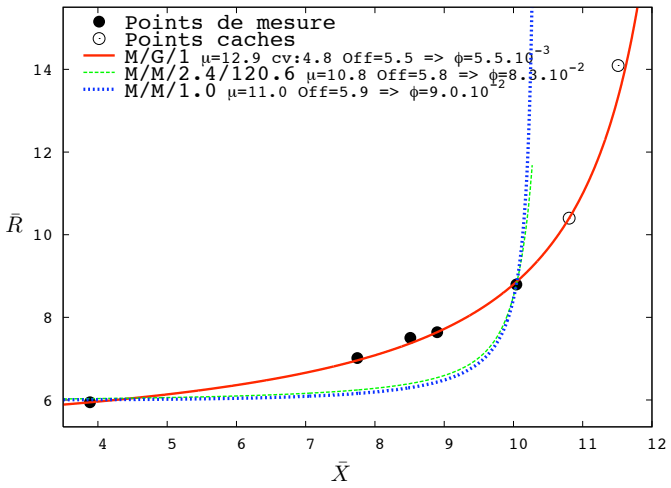


FIG. 4.3 – Un contrôleur disque - Jeu 2

Les Figures 4.2 et 4.3 illustrent les résultats obtenus respectivement pour le Jeu 1 et le Jeu 2. L'observation de la Figure 4.2 montre que le meilleur modèle parmi toutes les briques considérées pour le Jeu 1 est dérivé de la brique M/M/C/K, tandis que pour le Jeu 2 la Figure 4.3 indique que le modèle lauréat est une file M/G/1. Dans les deux cas, le modèle lauréat parvient à reproduire finement les points de mesure, et réussit également à prédire avec précision les points de mesure volontairement cachés. Pour le

Jeu 1 les erreurs relatives des prédictions délivrées par le modèle lauréat sont inférieures à 1% (à la fois celles sur le débit moyen et celles sur le temps de séjour moyen d'une requête E/S) et elles sont inférieures à 5% pour le Jeu 2. Notons que nous obtenons également des prédictions très précises si les points cachés sont choisis autrement.

Si l'on recherche un modèle constructif des performances d'E/S du contrôleur associé au Jeu 2 , il peut être utile de savoir que ni une file M/M/C, ni une file M/M/C/K ne semblent constituer de bons candidats pour modéliser le système. En effet, les résultats obtenus ici montrent que même avec la meilleure combinaison possible de leurs paramètres, aucune de ces files ne parvient à reproduire fidèlement les performances mesurées du contrôleur.

Le modèle proposé par notre approche peut également être utile dans le cadre d'une planification de capacité du système. En effet, le modèle lauréat permet de prédire quel sera le comportement du système en réponse à une hypothèse de croissance sur la charge de travail. En voici une illustration pour ces deux contrôleurs disques. Pour le jeu de mesures 1 (cf. Figure 4.2), avec une hypothèse de croissance de 10% sur le taux des arrivées des requêtes d'E/S (par rapport au niveau du dernier point de mesure), le modèle lauréat prévoit une dégradation du temps de réponse moyen du système de seulement 3%. A l'inverse, pour le jeu de mesures 2 (cf. Figure 4.3), une croissance de 10% du taux des arrivées devrait conduire selon le modèle lauréat à une augmentation de 290% du temps de réponse moyen du système.

4.4 Un réseau sans fil (IEEE 802.11)

Dans cet exemple, on s'intéresse à un réseau sans fil (de type IEEE 802.11) qui a été décrit et mesuré par Chandran-Wadia et al. [24]. La capacité de ce réseau WLAN s'élève à 2 Mbps et son comportement a été caractérisé pour deux tailles différentes de paquets : 500 et 1500 bytes. Chaque point de mesure indique le temps de séjour moyen d'un paquet dans le réseau WLAN et le débit moyen correspondant atteint par le réseau WLAN. On dispose donc de deux jeux de mesures : un pour chaque taille de paquet. Comme on peut s'y attendre, le comportement du réseau WLAN varie selon la taille des paquets. Etant donné que le temps de service d'un paquet dans ce réseau (comme celui de nombreux autres systèmes informatiques et réseaux de communication) comprend une partie fixe et incompressible, l'utilisation de petits paquets qui permet certes de réduire les délais de transfert des paquets, s'accompagne également d'une réduction de la bande passante maximale du réseau. Ce compromis entre réduction du délai et maximisation de la bande passante pour un réseau a fait l'objet de plusieurs études.

Afin de tenir compte de l'effet de la taille des paquets sur le réseau WLAN, on suppose que les temps de service moyens intrinsèques de toutes nos briques $1/\mu$ peuvent être exprimées comme :

$$\frac{1}{\mu} = S_0 + \frac{u}{capa} \tag{4.1}$$

où u représente la taille d'un paquet dans l'unité de traitement considérée (en bits ici), S_0 correspond à la partie fixe et incompressible du temps service exprimée ici comme un temps, et *capa* désigne la capacité de traitement du serveur en termes d'unité de travail par unité de temps. Les deux paramètre, S_0 et *capa*, doivent donc être déterminés afin de pouvoir définir le temps de service associé à chaque taille de paquets. Le fonctionnement de la méthode de calibrage reste le même sauf qu'au lieu d'avoir un paramètre indéterminé β_i pour le temps de service, on en a deux : S_0 et *capa*. Cette façon de représenter le temps de service peut certainement convenir à de nombreux autres systèmes comme par exemple les systèmes d'E/S, les systèmes de gestion de mémoire virtuelle ou encore les systèmes de gestion de fichiers. Toutefois, aussi générale soit-elle, l'utilisation d'une formule comme (4.1) suppose une certaine connaissance sur le système et peut d'une certaine manière s'apparenter à une approche semi-constructive.

La Figure 4.4 illustre les performances obtenues par ce réseau sans fil avec des paquets de 500 et de 1500 bytes. Notons que sur cette figure, le débit moyen du réseau est exprimé en nombre de requêtes transmises par unité de temps et non en bits (ou bytes) par unité de temps.

Nous observons qu'une file M/G/1 (avec $capa$ = 2.0E06, S_0 = 7.0E04 et γ = 8.4E-01) permet de reproduire finement le comportement du réseau WLAN pour les deux tailles de paquets. Comme précédemment, nous avons testé les capacités prédictives du modèle lauréat en laissant de côté un certain nombre de points de mesure lors de la recherche du modèle. Comme le montre la Figure 4.4, la file lauréate M/G/1 parvient à prédire avec précision les performances du réseau pour les points cachés. Au passage, soulignons que plus de 50% des points de mesure ont été cachés avant d'effectuer la recherche du modèle lauréat et que par ailleurs, les prédictions du modèle concernent entre autres les niveaux de charge les plus élevés du système pour lesquels la prédiction est probablement la plus difficile.

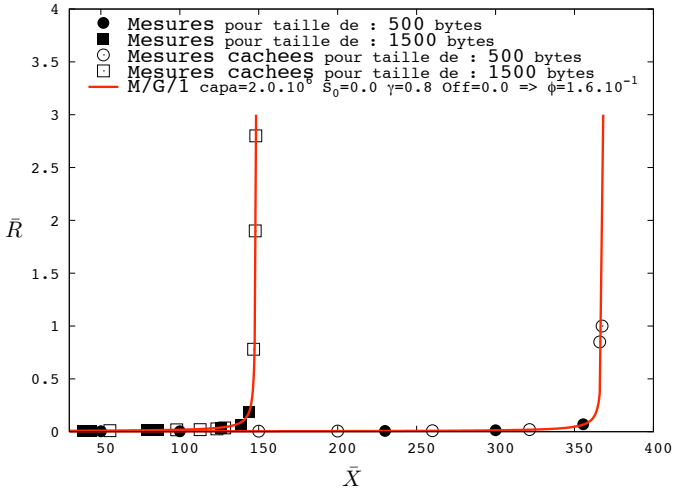


FIG. 4.4 – Un réseau sans fil

4.5 Un réseau Ethernet

Dans cet exemple, on considère le réseau Ethernet d'une capacité nominale égale à 10 Mbps qui a été décrit et mesuré par Wang et Keshav [103]. Les performances de ce réseau ont été mesurées pour trois tailles de paquets : 64, 512 et 1500 bytes. Ainsi, on dispose de trois jeux de mesures : un par taille de paquet. A chaque fois, un point de mesure associe un temps de séjour moyen d'un paquet dans le réseau et le débit moyen de paquets atteint par le réseau correspondant.

Le comportement du réseau Ethernet dépend naturellement de la taille des paquets, et comme pour l'exemple étudié dans la Section 4.4, nous prenons en compte cette dépendance dans nos briques en supposant que leur temps de service peut être exprimé selon la formule (4.1).

Premièrement, nous montrons qu'un modèle simple permet de représenter finement les performances mesurées de ce réseau Ethernet. Comme le montre la Figure 4.5, une simple file M/G/1 (avec $capa$ = 9.7E06, S_0 = 1.4E-04 et Off = 6.0E-01) permet de reproduire fidèlement le comportement mesuré du système à différents débits et pour différentes tailles de paquets.

Deuxièmement, nous montrons qu'en utilisant seulement deux des trois jeux de mesures, le modèle lauréat obtenu par la méthode HLM permet de prédire correctement les performances du réseau pour la troisième taille de paquets (correspondant au jeu non utilisé lors de la recherche du modèle). Par exemple, en mettant de côté le jeu de mesures pour les paquets de taille 64 bytes et en recherchant le modèle lauréat sur

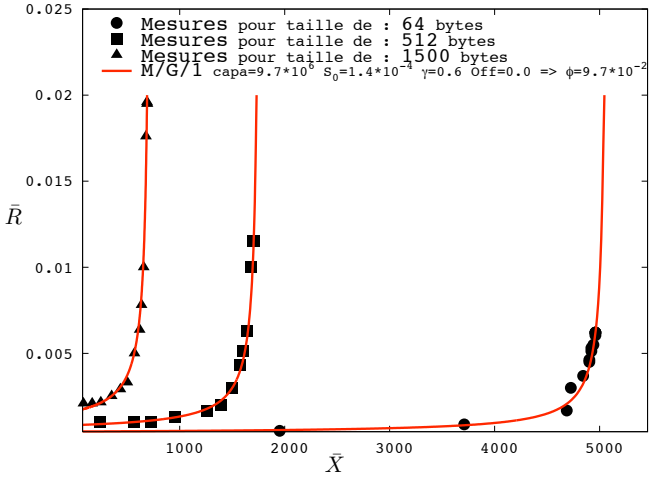


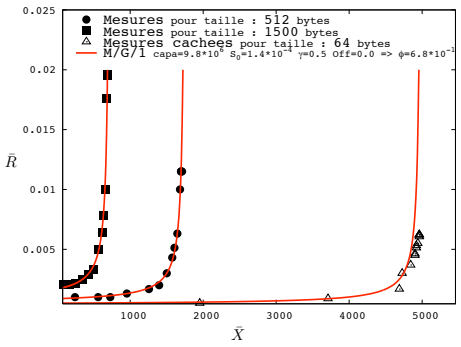
FIG. 4.5 – Un réseau Ethernet

les deux jeux de mesures restants, on obtient un modèle très proche de celui obtenu précédemment (i.e. une file M/G/1 avec $capa$ = 9.8E06, S_0 = 1.4E-04 et Off = 5.3E-01). Par conséquent, il n'est pas surprenant que le modèle trouvé parvienne à prédire correctement les performances du réseau lorsque la taille des paquets est de 64 bytes (cf. Figure 4.6(a)). On obtient des résultats similaires si l'on suppose que c'est le jeu de mesures pour les paquets de 512 bytes (respectivement 1500 bytes) qui est mis de côté comme l'illustre la Figure 4.6(b) (respectivement la Figure 4.6(c)).

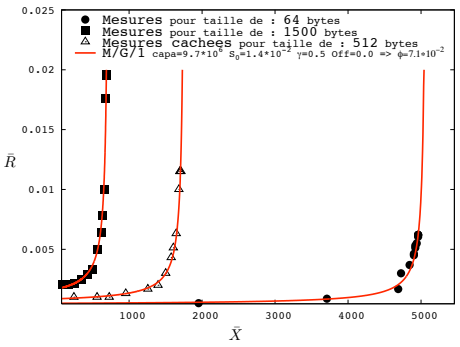
L'exécution rapide de notre méthode HLM[1] pour rechercher le modèle lauréat pourrait permettre de l'utiliser comme un outil efficace pour déterminer la taille optimale des paquets d'un réseau afin de satisfaire au mieux les besoins spécifiques d'une application (par exemple, une contrainte sur le délai moyen des paquets). Toutefois, cela suppose comme c'est le cas pour cet exemple, que le réseau peut être représenté par une des briques calibrées de notre méthode, et que nous ayons à notre disposition des jeux de mesures pour deux tailles différentes de paquets.

Enfin, soulignons que la dépendance du temps de service vis-à-vis de la taille des requêtes, que vise à représenter la formule (4.1), peut en réalité être bien plus complexe pour certains systèmes. Toutefois, le succès des prédictions de cet exemple et de l'exemple précédent (cf. Section 4.4) suggère, qu'au moins pour ces deux types de systèmes, cette formule simple convient.

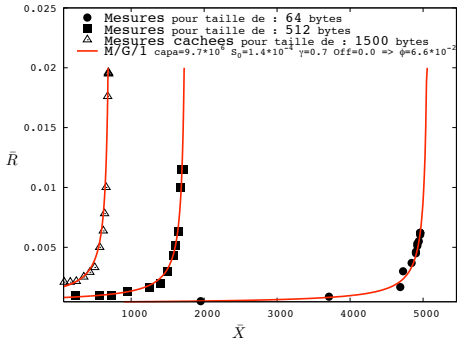
[1] typiquement entre quelques secondes et quelques minutes pour un processeur Intel Core 2 Duo de 2 GHz



(a) Avec seulement les jeux pour 512 et 1500 bytes



(b) Avec seulement les jeux pour 64 et 1500 bytes



(c) Avec seulement les jeux pour 64 et 512 bytes

FIG. 4.6 – Un réseau Ethernet

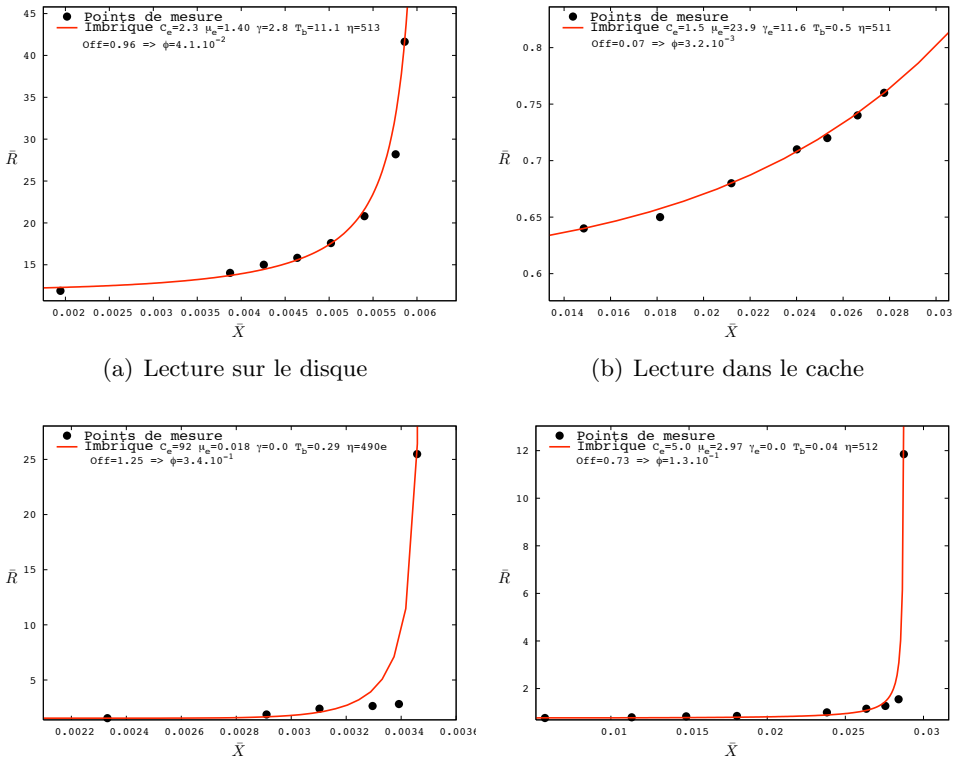
4.6 D'autres contrôleurs disques

A présent, nous nous intéressons à d'autres jeux de mesures obtenus également sur des contrôleurs disques d'un serveur central. Comme précédemment, les points de mesure indiquent le temps de séjour moyen d'une requête E/S en fonction du débit de sortie moyen atteint par le contrôleur. Les quatre jeux de mesures considérés dans cet exemple ont été obtenus sur le même contrôleur disque mais pour des charges de nature différente. Les deux premiers jeux de mesures, illustrés sur les Figures 4.7(a) et 4.7(b), représentent le comportement du contrôleur lorsque la charge soumise est composée uniquement de requêtes en lecture. Plus précisément, la Figure 4.7(a) correspond à une charge composée uniquement de requêtes auxquelles le contrôleur répond en allant consulter le contenu d'un disque tandis que pour la Figure 4.7(b), le contrôleur trouve la réponse directement dans son cache. Les Figures 4.7(c) et 4.7(d) correspondent à des

charges composées uniquement de requêtes disque en écriture dont le traitement se fait respectivement sur un disque et dans le cache.

La Figure 4.7 présente les résultats obtenus par notre méthode HLM pour ces quatre jeux de de mesures. Considérons le premier jeu de mesures représenté dans la Figure 4.7(a). A première vue, le comportement décrit par les points de mesure peut paraître similaire à ceux rencontrés jusqu'alors, et on peut penser qu'on ne devrait avoir aucune difficulté particulière pour trouver une brique qui le reproduise. Pourtant ce comportement est d'une simplicité trompeuse : aucune brique élémentaire, ni même la file pourtant réputée très générale M/G/C ne peut reproduire, quelle que soit son calibrage, un tel comportement. En effet, étant donné que toutes nos briques élémentaires ont un temps moyen de service qui reste fixe quel que soit le niveau de charge, il leur est impossible de reproduire, même très approximativement, à la fois le temps de réponse moyen du contrôleur à faible charge et son seuil de saturation (sauf à s'autoriser un offset négatif). Nous pensons que cette impossibilité tient au fait qu'en réalité le temps de service du contrôleur n'est pas fixe mais augmente à mesure que le débit atteint en sortie du contrôleur augmente. Comme le montre la Figure 4.7(a), seule la file imbriquée (définie dans le Chapitre 3, Section 3.2.4, page 71) permet de reproduire finement les performances décrites par les mesures avec le calibrage suivant : $T_b = 11.1$, $\eta = 513$, $\mu_s = 3.22$, $\gamma_s = 2.8$ et $\mathit{Off} = 0.96$. Les Figures 4.7(b) et 4.7(d) montrent des résultats similaires pour deux des autres jeux de mesures.

Le jeu de mesures représenté sur la Figure 4.7(c) présente une saturation encore plus abrupte que les autres. Ici aussi, aucune des briques élémentaires ne permet de reproduire même de façon très approximative les performances décrites par ces points de mesure. Seule la file imbriquée permet d'approcher relativement le comportement de ce jeu de mesures bien que subsiste un écart significatif. Une façon d'expliquer cet écart consiste à supposer que les mesures pour les plus hauts niveaux de débits sont imprécises ou qu'elles correspondent à un état du système qui ne soit pas stationnaire (les deux derniers points de mesure correspondent à une même valeur de débit mais à des niveaux de séjour moyens très différents). On peut également supposer qu'un autre type de brique doit être défini afin de mieux gérer ce type de comportement. Ceci étant dit, la brique imbriquée semble constituer un modèle adéquat car relativement simple et assez général pour représenter le comportement de systèmes pour lesquels ont lieu des élongations, même très rapides, du temps de service.



(a) Lecture sur le disque

(b) Lecture dans le cache

(c) Ecriture sur le disque

(d) Ecriture dans le cache

FIG. 4.7 – Un contrôleur disque pour quatre types de charge

4.7 Un système multiprocesseur

On s'intéresse à présent à un système d'exploitation dans lequel le nombre de processeurs physiques disponibles pour les processus change selon le niveau de charge du système. Pour ce système, chaque point de mesure indique un temps de réponse moyen pour les processus et le débit moyen correspondant atteint par le système (pour cette mesure). La Figure 4.8 illustre le jeu de mesures obtenu sur ce système. Nous observons que ce jeu de mesures présente une « discontinuité » puisque les temps de réponse moyen mesurés commencent par s'allonger à mesure que le débit augmente, puis chutent et enfin recommencent à s'allonger. Ainsi, il est clair qu'une brique « classique » comme une simple file M/M/C, M/M/C/K ou encore une file M/G/C, quelle que soit son calibrage, ne peut pas reproduire un tel comportement. Notre brique M/M/Cv (qui correspond à une file M/M/C pour laquelle le nombre de serveurs change si la charge dépasse un certain seuil) a été conçue spécialement pour ce type de systèmes. Comme le montre

la Figure 4.8, la file M/M/Cv parvient à reproduire assez fidèlement les performances mesurées du système.

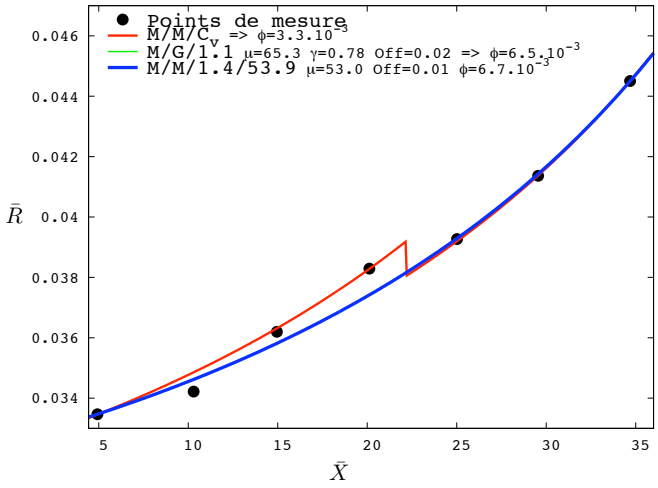


FIG. 4.8 – Les performances d'un système multiprocesseur

4.8 Un réseau maillé sans fil

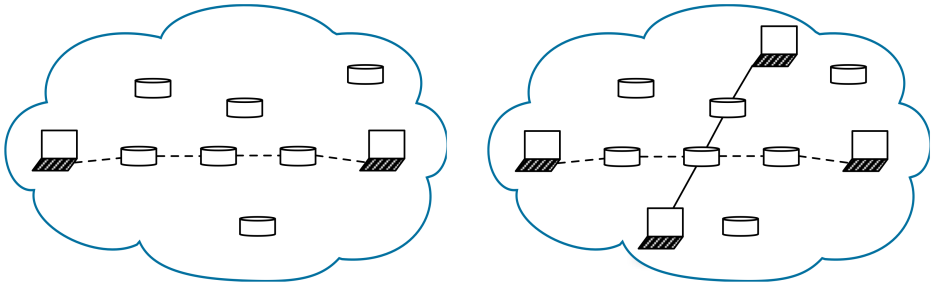
MeshDVNet

Dans cet exemple, nous nous intéressons aux performances d'un flot UDP dans un réseau maillé sans fil (« Wireless Mesh Network »). Le réseau maillé considéré, nommé MeshDVNet [88], est déployé dans les locaux du LIP6. Les clients du réseau se connectent à MeshDVNet en mode point d'accès (« Access Point ») via IEEE 802.11b/g tandis que les routeurs du réseau sont interconnectés via des liens IEEE 802.11a et opèrent tous sur le même canal radio. Tous les flots considérés dans cet exemple ont été engendrés par le générateur de trafic iperf [100]. Nous avons mesuré le débit moyen de chaque flot généré $\bar{X}$, correspondant au rapport entre le nombre total de paquets reçus et la durée de transmission (typiquement une minute), ainsi que le délai moyen d'acheminement d'un paquet $\bar{R}$, qui désigne le temps qui sépare en moyenne l'émission d'un paquet de sa réception par le destinataire[2].

Pour une taille de paquets donnée, nous désignons comme la *capacité d'un che-*

[2]Pour pouvoir mesurer $\bar{R}$, nous avons patché le programme iperf et déployé un réseau Fast Ethernet afin de synchroniser les horloges des clients sur un serveur NTPv4 ce qui permet une précision de l'ordre de la centaine de microsecondes [71].

min multi-saut entre une source et une destination le débit moyen maximal que peut transmettre un flot UDP.



(a) Config. A avec $n = 3$ routeurs (b) Config. B

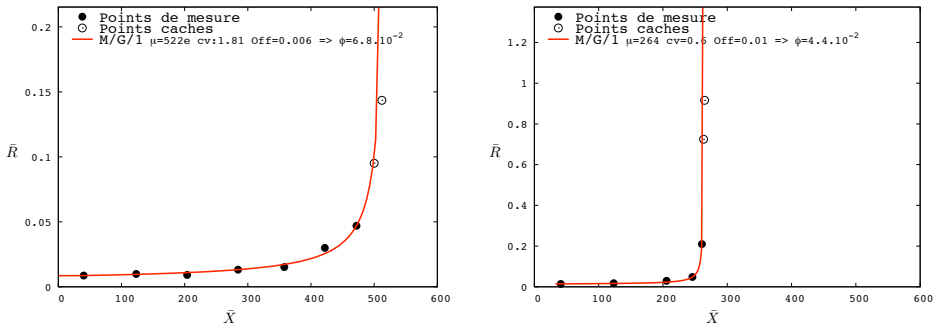
FIG. 4.9 – Les configurations considérées de MeshDVNet

Configuration A

Pour commencer, nous engendrons un seul flot UDP sur MeshDVNet, ce qui signifie qu'aucun autre trafic (mis à part le trafic de contrôle du réseau) ne perturbe son comportement. Le flot considéré traverse n routeurs sur son chemin, effectuant donc $n+1$ sauts avant d'atteindre sa destination. Cette situation correspond à la Configuration A de MeshDVNet, illustrée par la Figure 4.9(a). Le flot UDP est de type CBR (« Constant Bit Rate ») et la taille fixe de ses paquets est égale à 1520 bytes.

En faisant varier le taux d'émission des paquets depuis la source, on peut mesurer le comportement de ce flot pour différents niveaux de charge. Ainsi, dans cet exemple, chaque jeu de mesures indique le délai moyen d'acheminement d'un paquet $\bar{R}$ en fonction du taux d'arrivée des paquets à la destination $\bar{X}$. En changeant le chemin suivi par le flot UDP mesuré, nous avons pu obtenir plusieurs jeux de mesures, correspondants à des chemins et à un nombre de routeurs traversés différents. Les Figures 4.10(a) et 4.10(b) illustrent deux exemples de ces jeux de mesures pour respectivement $n = 2$ et $n = 4$.

D'abord, les jeux de mesures obtenus indiquent que la capacité d'un chemin dans un réseau maillé sans fil tend à décroître rapidement à mesure que le nombre de routeurs traversés n augmente, variant par exemple de 6.2Mbps (i.e. 510paquets/sec) pour $n = 2$ à 3.2Mbps (i.e. 265paquets/sec) pour $n = 4$. Cette décroissance rapide s'explique probablement par l'existence des contentions intra-flots dont les effets augmentent à mesure que n augmente.



(a) avec $n = 2$ (b) avec $n = 4$

FIG. 4.10 – Un seul flot UDP circule dans MeshDVNet

Ensuite, les résultats de la méthode HLM sur ces jeux de mesures montrent qu'une simple file M/G/1 munie d'un offset et adéquatement calibrée peut suffire à reproduire correctement les performances moyennes d'un flot UDP circulant seul sur un réseau maillé sans fil. Les Figures 4.10(a) et 4.10(b) illustrent ce résultat pour des chemins composés respectivement de $n = 2$ et $n = 4$ routeurs.

Enfin, les modèles lauréats délivrés par la méthode HLM, en plus de parvenir à reproduire finement les performances mesurées d'un flot UDP CBR circulant seul à travers un chemin donné de MeshDVNet, permettent de prévoir avec une bonne précision le comportement du flot pour des niveaux de charge (i.e. taux d'émission) non mesurés comme le montre la comparaison des prédictions délivrées par le modèle lauréat avec les points de mesure cachés (cf. Figures 4.10(a) et 4.10(b)). Notons que les points cachés correspondent volontairement aux débits les plus élevés car ils représentent certainement les niveaux pour lesquels les prédictions sont le plus difficiles à réaliser. Le seuil de saturation du modèle produit par la méthode HLM est probablement à rapprocher de la capacité du chemin considéré pour ces mesures.

Configuration B

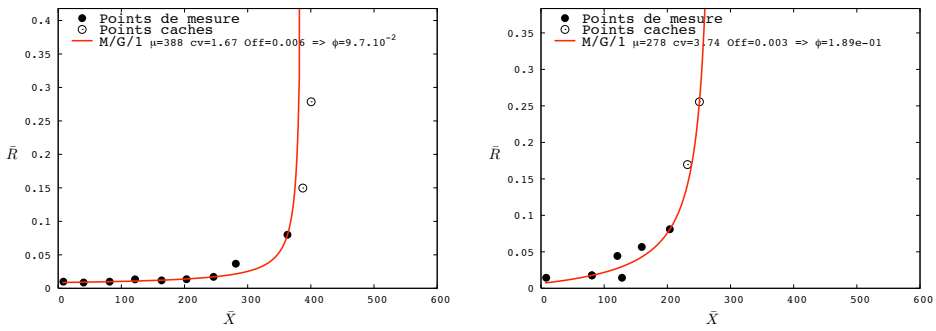
A présent on considère le cas où deux flots rivalisent pour l'accès à des ressources partagées du réseau maillé sans fil (canal radio, capacité des routeurs, etc.). Pour cela, on place MeshDVNet dans la Configuration B illustrée par la Figure 4.9(b) : on engendre deux flots UDP dont les chemins se croisent au niveau d'un routeur du réseau. Il est clair que l'apparition d'un flot sur un réseau maillé sans fil a des répercussions immédiates (en termes de collisions, de files, de délais et de pertes supplémentaires)

sur les performances des flots avec lesquels il rivalise pour l'accès aux ressources du réseau. Dans la configuration adoptée de MeshDVNet, le premier flot, désigné comme le *flot 1* et dont le chemin est représenté par la ligne en pointillé sur la Figure 4.9(b), traverse trois routeurs avant d'atteindre sa destination tandis que le deuxième flot, désigné comme le *flot 2* et dont le chemin est représenté par la ligne pleine sur cette figure, passe par deux routeurs. Notons que les deux flots UDP considérés transmettent du trafic CBR avec une taille fixe de paquets égale à 1520 bytes.

On s'intéresse aux performances du flot 1, en maintenant constant le taux d'émission de paquets du flot 2. Ici aussi, les jeux de mesure associent des délais moyens d'acheminement $\bar{R}$ à des niveaux moyens de débit $\bar{X}$ mesurés sur le flot 1.

Il est intéressant de remarquer que malgré la présence d'un flot concurrent, les performances moyennes $\bar{X}$ et $\bar{R}$ d'un flot UDP dans un réseau maillé restent reproductibles assez finement par un modèle aussi simple qu'une file M/G/1 munie d'un offset. Les Figures 4.11(a) et 4.11(b) représentent un jeu de mesures pour le flot 1 ainsi que le modèle lauréat proposé par la méthode HLM lorsque le taux d'émission de paquets du flot 2 est égal respectivement à 41 paquets/sec et 206 paquets/sec.

De plus, comme le montrent également les points cachés sur les Figures 4.11(a) et 4.11(b), les modèles lauréats de la méthode HLM permettent de prévoir avec précision le comportement du flot 1 pour d'autres niveaux de charge (à condition bien sûr que l'intensité du flot 2 reste la même).



(a) taux d'émission de 41 paquets/sec (b) taux d'émission de 206 paquets/sec

FIG. 4.11 – L'intensité du flot 2 est constante

Notons que la pertinence de certains points de mesure sur la Figure 4.11(b) peut être discutée. En effet, d'après notre connaissance du système, il est probable que la décroissance de $\bar{R}$ observée entre le troisième et le quatrième point de mesure corresponde plutôt à une incertitude sur ces mesures (qui aurait peut-être pu être évitée en

moyennant les performances mesurées sur des intervalles de temps plus longs) qu'à un aspect réel du comportement de MeshDVNet. Il est toutefois intéressant de noter que, malgré ces mesures entachées de « doute », la méthode HLM parvient à obtenir un modèle qui permet de prédire avec une précision raisonnable les performances du flot UDP mesuré à des niveaux de charge non mesurés[3].

4.9 Conclusion

Les jeux de mesures considérés dans ce chapitre sont issus de systèmes informatiques et de réseaux de communication réels et variés. Ces exemples corroborent le principe fondamental de la méthode HLM en montrant qu'un ensemble limité de modèles génériques relativement simples, munis d'un offset et adéquatement calibrés peut suffire à reproduire correctement le comportement d'entrée/sortie de multiples systèmes.

Nous terminons ce chapitre en rappelant brièvement les trois principales applications possibles de la méthode HLM que nous avons vues au cours des chapitre 3 et 4. Premièrement, l'utilisation de la méthode sur un sous-système appartenant à un système plus vaste peut permettre (à condition de disposer d'un jeu de mesures pour le sous-système) de trouver un modèle simple pour le sous-système qui pourra ensuite être incorporé dans une modélisation du système global. Cet usage de la méthode est à rapprocher des méthodes de décomposition existantes [16].

Deuxièmement, le modèle proposé par la méthode HLM peut être utile pour prédire les performances du système considéré à des niveaux de charge pour lesquels on ne dispose pas de mesures. Ces prédictions peuvent par exemple avoir pour but de savoir si le système concerné parviendra à satisfaire des objectifs de qualité de service si ses conditions de fonctionnement viennent à changer. En se basant sur une hypothèse de croissance de la charge soumise au système, le modèle lauréat de la méthode HLM permet de savoir, rapidement, simplement et sans avoir à analyser le fonctionnement intérieur du système, si ces objectifs seront tenus ou pas. Un autre exemple d'utilisation possible concerne la prévision des performances d'un système si la taille de ses clients (i.e. requêtes) vient à changer.

Enfin, les résultats obtenus par la méthode HLM peuvent permettre de révéler des propriétés inattendues d'un système telle une élongation importante de son temps de service lorsque le niveau de la charge augmente ou encore l'apparition d'un goulet d'étranglement qui réduit le degré de parallélisme prévu du système. En ce sens, nous

[3] On aurait pu tenir compte de ces incertitudes en réduisant l'influence de ces mesures sur le calibrage recherché du modèle grâce aux coefficients pondérateurs w_i présentés dans le Chapitre 2

pensons que la méthode HLM peut être utile et complémentaire à une modélisation constructive du système car son utilisation peut permettre dans certains cas de découvrir des propriétés du système difficilement décelables depuis sa structure interne. Toujours dans ce même ordre d'idée, l'utilisation de la méthode HLM peut permettre, lorsqu'elle est appliquée à des systèmes théoriques de type file d'attente, de guider la recherche d'une approximation du système considéré en indiquant rapidement quel type de briques peut ou ne peut pas fonctionner (cf. Chapitre 3, Section 3.3, page 79).

Il existe probablement d'autres applications possibles de la méthode HLM. Nous en décrivons succinctement deux que nous n'avons pas encore eues le temps d'expérimenter : le dimensionnement d'un système en conception et la modélisation à la volée d'un système pour lequel on a des mesures en-ligne. On considère d'abord le cas d'un système en cours de conception pour lequel on dispose d'un modèle. En adoptant ce modèle comme unique brique de la méthode et en exprimant les objectifs de performances recherchés pour ce systèmes sous la forme d'un jeu de mesures, la méthode HLM pourrait peut être permettre de déterminer un calibrage des paramètres du système qui permet d'atteindre ces objectifs. Concernant la modélisation à la volée, nous pensons que l'exécution rapide de la méthode HLM la rend adaptée à un contexte dynamique dans lequel les mesures pourraient être reçues de façon continue. La recherche du modèle se ferait donc en-ligne et pourrait servir par exemple à détecter une anomalie sur le système si à un instant donné une nouvelle mesure contrarie le modèle obtenu par les points de mesure précédents. L'expérimentation de ces deux idées sur des exemples réels fait partie des priorités pour les travaux futurs relatifs à la méthode HLM.

Chapitre 5

Effets des propriétés distributionnelles d'ordre supérieur

Sommaire

5.1 Introduction

Ce chapitre rapporte des résultats des travaux menés conjointement avec le Professeur Brandwajn. Tous les résultats présentés dans ce chapitre, sauf mention contraire, ont été obtenus par des méthodes exactes de résolution numérique [20, 17]. Ces résultats, bien que décorrélés des chapitres précédents, ont été obtenus en envisageant l'ajout d'une brique supplémentaire, la file M/G/C, à la méthode de modélisation HLM. Nous avons d'abord remarqué que les deux premiers moments de la distribution du temps de service d'une file M/G/C ne suffisent pas à déterminer sans ambiguïté ses paramètres de performances moyens avant d'approfondir et d'étendre ces résultats à d'autres modèles que nous présentons ici.

Le degré de description de la distribution du temps de service d'un modèle de type file d'attente est fonction des objectifs poursuivis par le modèle et de la file considérée. Par exemple, pour évaluer les paramètres de performances moyens d'une file M/G/1, il suffit de définir la valeur des deux premiers moments de son temps de service (cf. formule de Pollaczek-Khintchine [56, 2, 5]). En revanche, si l'on s'intéresse au calcul exact du second moment du temps d'attente dans une file M/G/1, on doit alors considérer les trois premiers moments de la distribution du temps de service [26].

Pour déterminer la distribution stationnaire du nombre de clients dans une file M/G/C, la plupart des méthodes numériques existantes [70, 42, 99, 94] considère une distribution de type Phase ou de type Cox, ou plus spécifiquement de type Erlang dans le cas où le coefficient de variation du temps de service est inférieur à 1, pour représenter la distribution du temps de service des serveurs. Il faut donc reproduire les propriétés de la distribution réelle du temps de service dans la *distribution de remplacement*. En théorie, cette reproduction est toujours possible puisque toute distribution (à quelques restrictions près [2, 22, 91]) peut être approchée « d'aussi près que l'on veut » par une loi de type Phase ou par une loi en Cox. Cependant, il semble que peu de travaux ont tenté de déterminer quels aspects de la distribution réelle du temps de service doivent être représentés dans la distribution de service de remplacement afin de préserver l'intégrité des résultats obtenus par le modèle. On peut par exemple s'interroger si la prise en compte des deux premiers moments du temps de service est suffisante pour déterminer les paramètres de performances moyens d'une file M/G/C, comme c'est le cas pour la file M/G/1 et comme semble également le suggérer la quasi-totalité des approximations existantes [13, 15, 45, 80, 40, 98, 73, 54, 55].

Dans ce chapitre, nous mettons en évidence les effets parfois méconnus des propriétés distributionnelles d'ordre supérieur du temps de service et du temps d'inter-arrivées

pour des files classiques de la théorie des files d'attente comme les files M/G/1, M/G/C et G/M/C. Nous montrons notamment la nette influence que peuvent avoir, au-delà de leurs deux premiers moments, les distributions du temps de service et du temps d'inter-arrivées sur la distribution du nombre de clients dans la file, et en particulier sur sa valeur moyenne sauf bien entendu dans le cas d'une file M/G/1 à capacité d'accueil illimitée.

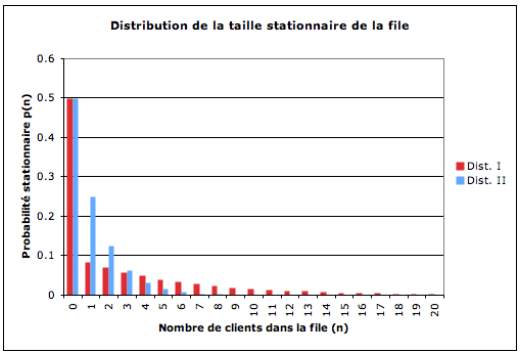
Pour décrire les propriétés d'ordre supérieur d'une distribution, on fait souvent appel aux coefficient de variation, de dissymétrie (« skewness ») et d'aplatissement (« kurtosis ») [95]. De la même manière que le coefficient de variation se rapporte aux deux premiers moments d'une distribution, le coefficient de dissymétrie (respectivement le coefficient d'aplatissement) se rapporte aux trois premiers moments (respectivement aux quatre premiers moments) de la distribution.

5.2 Cas d'une file monoserveur

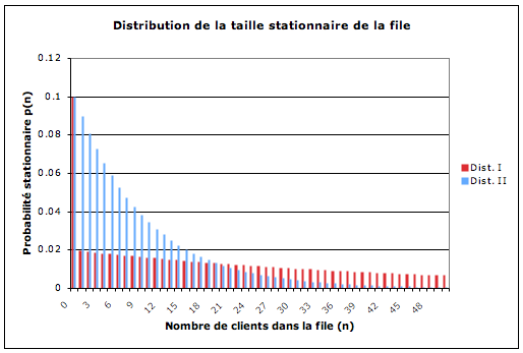
La distribution de la taille d'une file M/G/1 dépend sensiblement des moments d'ordre supérieur du service

En régime stationnaire, il est bien connu que le nombre moyen de clients dans une file M/G/1 à capacité d'accueil illimitée dépend seulement des deux premiers moments de la distribution du temps de service comme l'indique la formule de Pollaczek-Khintchine [56, 2, 5]. Il est également connu [26] que les trois premiers (respectivement, les quatre premiers) moments de la distribution du temps de service doivent être connus pour calculer le second (respectivement, le troisième) moment du temps d'attente. En revanche, il est peut être moins connu que la distribution du nombre de clients dans une file M/G/1 peut être très sensible aux propriétés d'ordre supérieur de la distribution du temps de service. La Figure 5.1 illustre cette dépendance en comparant la distribution du nombre de clients dans une file M/G/1 pour deux distributions différentes du temps de service ayant les deux mêmes premiers moments. Les temps de service considérés dans cet exemple sont représentés par des distribution en Cox-2 dont les paramètres sont indiqués dans la Table 5.1. Par exemple, les distributions Dist. I et Dist. II correspondent à un coefficient de variation égal à 3 mais avec des propriétés distributionnelles d'ordre supérieur, comme le coefficient de dissymétrie et le coefficient d'aplatissement, différentes.

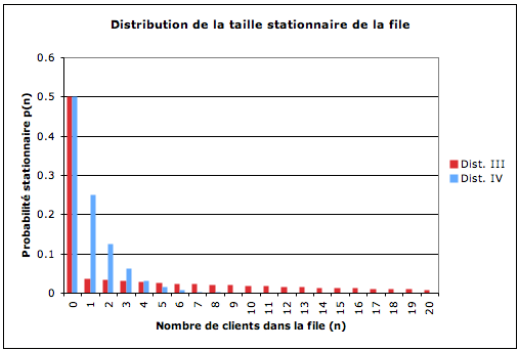
Deux remarques peuvent être faites d'après les résultats de la Figure 5.1. D'une part, ces résultats indiquent que les effets des propriétés d'ordre supérieur de la distribution du temps de service d'une file M/G/1 peuvent être « visibles » à des niveaux d'utilisation



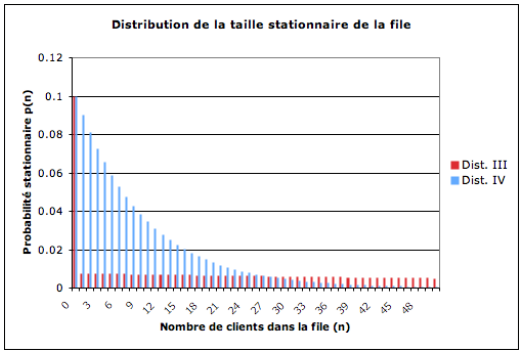
(a) Coefficient de variation : 3, utilisation du serveur : 0.5



(b) Coefficient de variation : 3, utilisation du serveur : 0.9



(c) Coefficient de variation : 5, utilisation du serveur : 0.5



(d) Coefficient de variation : 5, utilisation du serveur : 0.9

FIG. 5.1 – L'effet de la distribution du temps de service sur le nombre de clients dans une file M/G/1

modérés d'un serveur. Il est par exemple intéressant d'observer que pour un coefficient de variation égal à 3 et une utilisation du serveur égale à 0.5, la probabilité d'avoir plus de 20 clients dans la file (un peu plus que trois fois la valeur moyenne) est environ de 0.1% dans un cas, et d'un ordre de grandeur supérieure pour une autre distribution du temps de service avec pourtant les deux mêmes premiers moments. D'autre part, ces résultats suggèrent que les effets des propriétés d'ordre supérieur à 2 de la distribution du temps de service tendent à s'intensifier à mesure que l'utilisation du serveur et que le coefficient de variation de la distribution du temps de service augmentent.

Cette dépendance peut par exemple avoir des conséquences pour le dimensionnement de la taille des tampons (« buffers »). En supposant un processus d'arrivée poissonien, une approche rapide pour décider de la taille des tampons pour un système donné consiste à la fixer comme étant égale à un certain multiple (par exemple trois

Distribution	Temps de service moyen	Coefficient de variation	Coefficient de dissymétrie	Coefficient d'aplatissement	Taux de service de l'étage 1	Probabilité d'aller à l'étage 2	Taux de service de l'étage 2
Dist. I	1	3	4.5	27.3	1.00E04	2.00E-01	2.00E-01
Dist. II	1	3	3.56E03	1.90E07	1.0	2.50E-07	2.50E-04
Dist. III	1	5	7.5	75.1	1.00E04	7.69E-02	7.69E-02
Dist. IV	1	5	6.91E03	6.63E07	1.0	8.33E-08	8.33E-05

TAB. 5.1 – Les paramètres et les propriétés des distributions du temps de service utilisées dans la Figure 5.1

ou six fois) du nombre moyen de clients dans une file M/G/1 [72]. Cela revient à dimensionner la taille des files en s'appuyant uniquement sur les deux premiers moments de la distribution du temps de service (cf. formule de Pollaczek-Khintchine). Or étant donnée l'importante dépendance de la distribution stationnaire du nombre de clients dans une file M/G/1 vis-à-vis des propriétés d'ordre supérieur de la distribution du temps de service, une telle approche peut conduire à des dimensionnements inadaptés.

Pour une file M/G/1/K, les performances moyennes sont très dépendantes des moments d'ordre supérieur du service

L'utilisation d'une file M/G/1/K, c'est-à-dire une file M/G/1 avec une capacité d'accueil limitée, constitue naturellement une façon plus directe de dimensionner la taille des tampons (« buffers »). Même s'il semble qu'il existe moins de résultats théoriques pour cette file que pour la file M/G/1, il est toutefois connu que la distribution stationnaire d'une file M/G/1/K peut être obtenue à partir de celle d'une file à capacité illimitée M/G/1 en renormalisant les probabilités d'état aux instants de départ de la file M/G/1 associée aux seuls états possibles, i.e. de 0 à $K-1$ [52, 37, 27]. Notons que cette approche ne peut naturellement fonctionner que si le taux des arrivées n'excède pas le taux de service car sinon la file M/G/1 à capacité illimitée associée serait instable.

Bien que la distribution du nombre de clients dans une file M/G/1/K puisse être obtenue à partir de celle d'une file M/G/1 à capacité illimitée, et que pour cette dernière le nombre moyen de clients dans la file dépende uniquement des deux premiers moments de la distribution du temps de service, il en va autrement pour le nombre moyen de clients dans la file M/G/1/K. L'exemple présenté dans la Table 5.2 montre que même les trois premiers moments de la distribution du temps de service ne suffisent pas à déterminer sans ambiguité le nombre moyen de clients dans une file M/G/1/K. En effet, dans cet exemple, les valeurs obtenues pour le nombre moyen de clients dans la file diffèrent de plus de 27% alors que les deux distributions en Cox-3 considérées pour le temps de service ont les trois mêmes premiers moments.

	Première Cox-3	Seconde Cox-3	Différence relative
Taux des arrivées	1		-
Capacité d'accueil de la file	30		-
Temps de service moyen	1		-
Coefficient de variation	6.40		-
Coefficient de dissymétrie	2.33E03		-
Coefficient d'aplatissement	7.43E06	1.44E07	-
Nombre moyen de clients dans la M/G/1/K	3.98	5.07	27.4 %

TAB. 5.2 – L'effet des propriétés au-delà du troisième moment sur le nombre moyen de clients dans une file M/G/1/K

Etant donné que le nombre moyen de clients dans une file M/G/1 à capacité d'accueil illimitée dépend uniquement des deux premiers moments de la distribution du temps de service, et que pour une file M/G/1/K avec K=1 il n'y a aucune dépendance distributionnelle (puisqu'il n'y a pas d'attente), il est intéressant d'observer comment varie la dépendance des propriétés d'ordre supérieur avec K, i.e. la capacité d'accueil de la file. La Figure 5.2 compare la probabilité stationnaire d'avoir exactement un client dans la file pour les distributions Dist. I et Dist. II de la Table 5.1. Nous avons également représenté sur cette figure la différence relative pour la probabilité d'avoir K clients dans la file, c'est-à-dire d'avoir le tampon (« buffer ») de la file plein. Nous observons que bien que les deux premiers moments de la distribution du temps de service soient identiques, les propriétés d'ordre supérieur conduisent les probabilités considérées à des valeurs très différentes. Concernant la probabilité d'avoir le buffer plein, nous remarquons que bien que la différence relative pour ces deux distributions tende à se réduire à mesure que la taille du tampon K augmente, cette différence reste relativement élevée même pour des valeurs assez grandes de K.

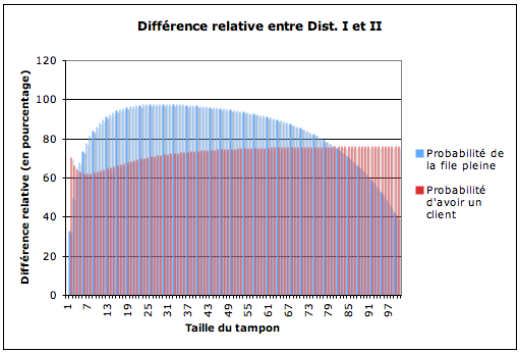
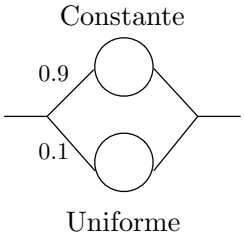


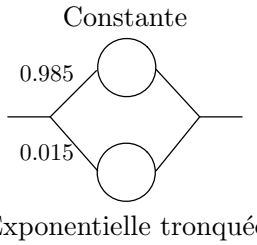
FIG. 5.2 – La différence relative des probabilités d'état pour les distributions Dist. I et Dist. II en fonction de la taille du tampon d'une file M/G/1/K

Afin de mieux illustrer l'influence des propriétés d'ordre supérieur de la distribution

du temps de service sur le comportement d'une file M/G/1/K, nous considérons les performances en lecture de deux équipements simples de stockage de données munis d'un cache et d'un tampon (« buffer ») de taille égale à 10, i.e. K = 10, et pour lesquels les arrivées sont supposées poissoniennes. Lorsque l'information demandée se trouve dans le cache, il s'agit d'un « hit » et le temps de service est alors représenté par une constante (en supposant des enregistrements de taille fixe). A l'inverse, lorsque l'information n'est pas dans le cache, on parle alors de « miss » et l'information doit être récupérée à partir du dispositif physique sous-jacent au système de stockage de données. Dans un cas, nous représentons le temps de service du dispositif physique sous-jacent, c'est-à-dire le temps de service des « miss », par une distribution uniforme tandis que dans l'autre cas, ce temps de service est représentée par une exponentielle tronquée [49]. La Figure 5.3 synthétise la description du temps de service pour les deux équipements considérés. Nous avons décidé des valeurs respectives du temps de service des « hit », des « miss » et de leurs proportions pour ces deux systèmes de stockage de données de façon à ce que les moyennes et les coefficients de variation de leur temps de service soient identiques, comme l'indique la Table 5.3.



(a) Loi uniforme pour le service des « miss »



(b) Loi exponentielle tronquée pour le service des « miss »

FIG. 5.3 – Deux distributions différentes du temps de service avec les deux premiers moments identiques

La Table 5.3 contient également les résultats obtenus par simulation pour ces deux systèmes de stockage de données. Les intervalles de confiance ont été estimés à 95%. Nous nous intéressons au débit de sortie maximal pouvant être atteint par ces deux systèmes tout en maintenant le temps moyen de séjour dans le système inférieur à 5 ms. Les résultats montrent que les débits moyens de sortie des deux systèmes diffèrent de plus de 20% alors que le coefficient de variation du temps de service pour les deux systèmes est seulement de 1.6.

	Temps de service des « miss » : loi uniforme	Temps de service des « miss » : loi exponentielle tronquée	Différence relative
Probabilité de « hit »	0.9	0.985	-
Temps de service des « hit »	1	1.64	-
Temps de service des « miss »	Uniforme [2,18]	Exponentielle tronquée ; moyenne : 20, max : 100	-
Temps moyen de service	1.9		-
Coefficient de variation	1.62		-
Débit moyen de sortie pour un temps moyen de séjour de 5ms	0.257±0.001	0.312±0.001	21.4 %

TAB. 5.3 – Débit moyen de sortie pour deux équipements de stockage ayant les deux mêmes premiers moments de temps de service

D'après notre expérience, il semble que l'influence des propriétés d'ordre supérieur tende à s'accroître à mesure que le coefficient de variation et le coefficient de dissymétrie du temps de service augmentent. Or il existe de nombreux systèmes informatiques pour lesquels la valeur de ces deux coefficients peut être élevée. En plus des systèmes de stockage dotés d'un cache, c'est également le cas des processeurs modernes pour lesquels le temps d'exécution d'une instruction d'un programme peut être très irrégulier : les instructions les plus fréquentes sont très optimisées, celles moins fréquentes sont plus lentes à exécuter, et enfin certaines instructions encore plus rares, implantées comme des appels à des sous-routines, peuvent conduire à des temps d'exécution supérieurs de plus d'un ordre de grandeur.

Les effets des propriétés d'ordre supérieur du temps de service peuvent certainement être aussi observés sur les réseaux du fait de la distribution irrégulière de la taille des paquets. Nous considérons à présent la probabilité qu'une file située dans un réseau optique, modélisée par une file M/G/1/K et pouvant contenir 10 messages (K=10), soit pleine. Nous supposons que les paquets entrants dans la file peuvent être de trois tailles différentes. Dans le premier cas, qui s'apparente au trafic IP mesuré dans certains réseaux actuels, les tailles des paquets sont de 40, 300 et 1500 bytes avec respectivement des probabilités de 0.5, 0.3 et 0.2. Dans le second cas, des paquets plus longs sont utilisés : 150, 500 et 5000 bytes, avec respectivement des probabilités de 0.426, 0.561 et 0.013. Les deux distributions considérées pour la taille des paquets conduisent à la même valeur moyenne de 410 bytes et au même coefficient de variation de 1.36, mais à des propriétés d'ordre supérieur différentes. Le temps que met la file à servir un paquet est représenté par une constante dont la valeur est proportionnelle à la taille du paquet. Pour un taux d'arrivée des paquets égal au taux de service de la file, les résultats de la

simulation indiquent que les probabilités d'avoir la file pleine diffèrent d'environ 20% selon le type de trafic considéré (12.5% dans un cas et 10.5% dans l'autre) alors que dans les deux cas, les distributions du temps de service ont les deux mêmes premiers moments.

Ces dépendances sont également très fortes sur les performances d'un réseau fermé, même très simple

A présent, nous considérons le cas d'une file M/G/1 placée dans un réseau fermé très simple. Le réseau considéré comporte deux noeuds correspondant chacun à une file monoserveur comme l'illustre la Figure 5.4. Le temps de service dans le Noeud 1 est supposé général tandis que celui du Noeud 2 est distribué exponentiellement. N clients (i.e. jetons) se répartissent entre les deux noeuds. Rappelons que les performances pour ce réseau ont été obtenues par une méthode exacte de récurrence [20].

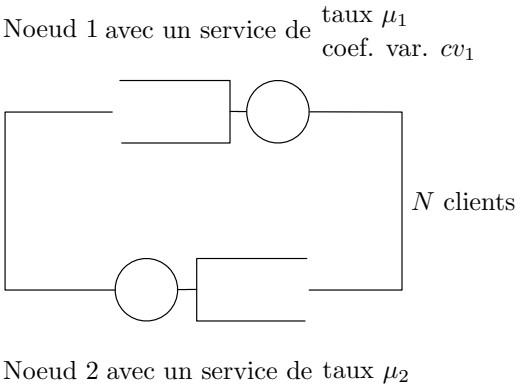
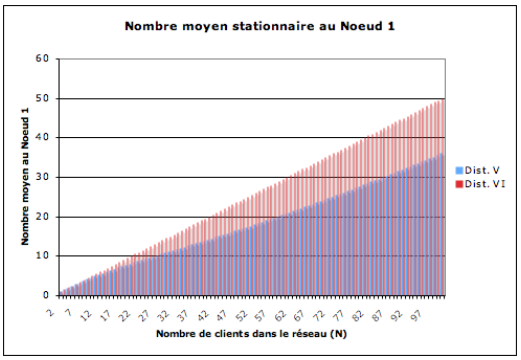


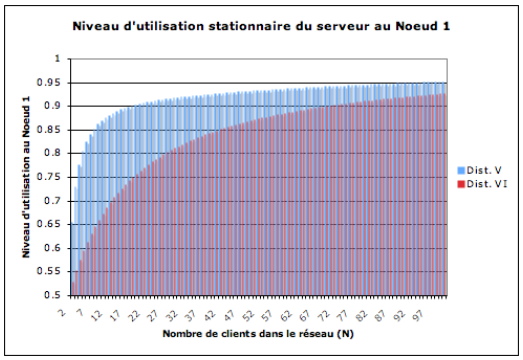
FIG. 5.4 – Une file M/G/1 en réseau avec une file M/M/1

Dans cette section, nous nous intéressons uniquement au comportement du Noeud 1 et plus précisément à l'influence des propriétés d'ordre supérieur de la distribution de son temps de service sur ses performances. Dans un premier temps, nous supposons que le taux de service du Noeud 2 est égal à 1, c'est-à-dire $\mu_2 = 1$.

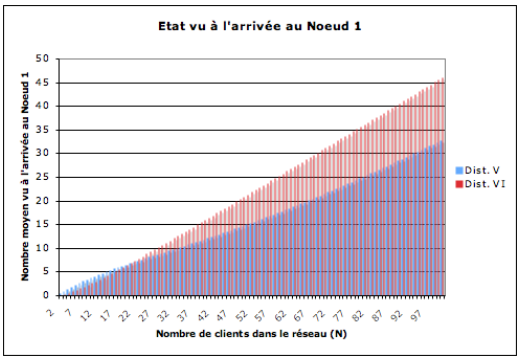
Nous avons représenté sur la Figure 5.5(a) le nombre moyen de clients au Noeud 1 en fonction du nombre total de clients dans le réseau N pour deux distributions en Cox-2 : Dist. V et Dist. VI ayant les deux mêmes premiers moment mais des propriétés d'ordre supérieur différentes. Ces deux distributions sont décrites dans la Table 5.4, et ont une valeur moyenne égale à 1 et un coefficient de variation égal à 4. Les Figures 5.5(b) et 5.5(c) montrent elles respectivement le taux d'utilisation du serveur du Noeud 1 et le nombre moyen de clients vu à l'arrivée au Noeud 1 obtenus pour ces deux distributions.



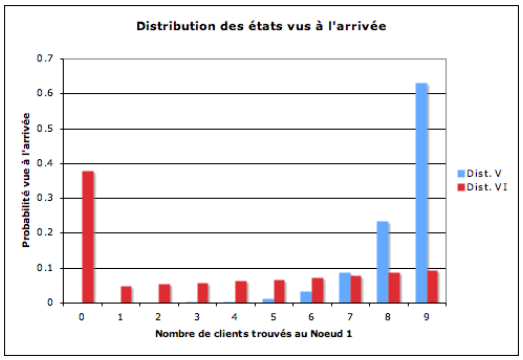
(a) Influence sur le nombre moyen de clients dans la file



(b) Influence sur le niveau d'utilisation du serveur de la file



(c) Influence sur le nombre moyen de clients vu à l'arrivée



(d) Influence sur probabilités d'états vues à l'arrivée

FIG. 5.5 – L'effet des propriétés d'ordre supérieur du temps de service du Noeud 1 sur ses performances

Nous observons sur les Figures 5.5(a), 5.5(b) et 5.5(c) l'effet important des propriétés d'ordre supérieur de la distribution du temps de service du Noeud 1 sur le nombre moyen de clients à tout instant dans ce noeud, sur le niveau d'utilisation de son serveur et sur le nombre moyen de clients vu à l'arrivée dans ce noeud. Ainsi, pour N=10, selon la distribution considérée, la différence sur le taux d'utilisation du serveur atteint près de 30%, et la différence est également de 30% lorsqu'on compare le nombre moyen de clients dans la file pour N=30.

A présent, on s'intéresse plus particulièrement aux états vus à l'arrivée dans le Noeud 1. Pour cela, le nombre total de clients dans le réseau est fixé à $N = 10$ et le temps de service moyen au Noeud 2, $1/\mu_2$, est égal à 1/3. La Figure 5.5(d) représente les probabilités d'état trouvées à l'arrivée pour le Noeud 1 en considérant les deux distributions Dist. V et Dist. VI. Il est intéressant de noter que la distribution des états

Distribution	Temps de service moyen	Coefficient de variation	Coefficient de dissymétrie	Coefficient d'aplatissement	Taux de service de l'étage 1	Probabilité d'aller à l'étage 2	Taux de service de l'étage 2
Dist. V	1	4	54.10	4.11E03	1.11	1.3E-03	1.32E-02
Dist. VI	1	4	6.01	4.83E01	1.00E03	1.17E-01	1.18E-01

TAB. 5.4 – Les paramètres des distributions du temps de service du Noeud 1

vus à l'arrivée est radicalement différente selon la distribution du temps de service considérée pour le Noeud 1. En particulier, si l'on considère la probabilité qu'un client entrant dans le Noeud 1 n'ait pas besoin d'attendre avant de pouvoir être servi, nous observons qu'elle varie de presque 0% dans un cas à près de 40% dans l'autre cas.

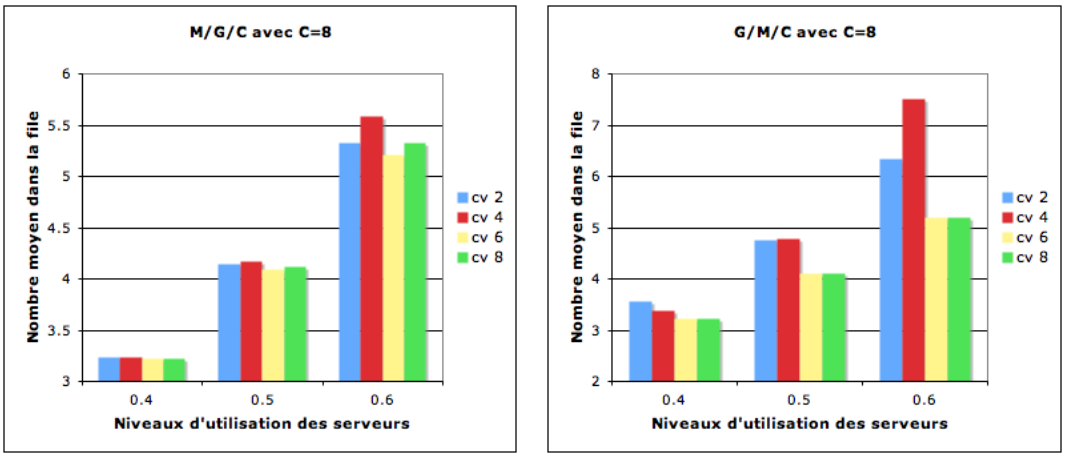
Ces résultats montrent que pour un réseau fermé très simple avec uniquement un seul serveur non-exponentiel, les performances du système peuvent être sensiblement dépendantes des propriétés d'ordre supérieur de la distribution du temps de service, au-delà de ses deux premiers moments. A la lumière de ces résultats, il est possible que les conditions d'applicabilité de nombreuses approximations pour les réseaux fermés non-exponentiels doivent être ré-évaluées [68, 82, 89, 102, 108, 41].

5.3 Cas des files multiserveurs

Des comportements « contre-intuitifs »

L'influence des propriétés d'ordre supérieur des distributions du temps de service ou des inter-arrivées pour les files multiserveurs est telle qu'elle peut déboucher sur des comportements « contre-intuitifs ». Nous considérons quatre distributions du temps de service à valeur moyenne identique mais dont les coefficients de variation s'échelonne entre 2 et 8. Ces distributions sont de type Cox-2 et sont décrites en détail dans la Table 5.5. Nous considérons chacune d'entre elles comme distribution du temps de service pour une file M/G/C avec 8 serveurs. La Figure 5.6(a) représente le nombre moyen de clients dans la file obtenu pour ces quatre distributions. On pourrait s'attendre à ce qu'en augmentant le coefficient de variation du temps de service, le nombre moyen de clients dans la file augmente également, comme c'est le cas pour une file M/G/1. Or l'observation de la Figure 5.6(a) montre que le nombre moyen de clients dans une file M/G/C obtenu pour une distribution donnée de son temps de service peut être inférieur à celui obtenu pour une autre distribution avec un coefficient de variation inférieur. Ces résultats suggèrent clairement l'importance des moments d'ordre supérieur de la distribution du temps de service pour une file M/G/C que nous montrons plus nettement ci-dessous. La Figure 5.6(b) montre que des résultats semblables peuvent être obtenus

pour une file G/M/C. Notons que des comportements « contre-intuitifs » similaires ont été également décrits pour divers systèmes d'attente [107, 104, 14, 66, 106, 39].



(a) pour une file M/G/C (b) pour une file G/M/C

FIG. 5.6 – Des comportements « contre-intuitifs » dus aux propriétés distributionnelles d'ordre supérieur

Précisons que dans le cas de la file G/M/C, les taux des étages des Cox-2 du processus d'arrivée ont été proportionnellement ajustés de façon à pouvoir considérer le comportement de la file à plusieurs niveaux d'utilisation de ses serveurs. Rappelons également que tous les résultats de cette section ont été obtenus par une méthode exacte de résolution numérique basée sur les probabilités conditionnelles [17].

Distribution	Temps de service moyen	Coefficient de variation	Coefficient de dissymétrie	Coefficient d'aplatissement	Taux de service de l'étage 1	Probabilité d'aller à l'étage 2	Taux de service de l'étage 2
cv 2	1	2	3.1	12.8	1.00E03	4.00E-01	3.99E-01
cv 4	1	4	14.3	280.2	1.67	5.06E-02	2.03E-02
cv 6	1	6	86.0	1.00E04	1.11	5.68E-03	5.68E-04
cv 8	1	8	117.0	1.85E04	1.11	3.17E-03	3.16E-04

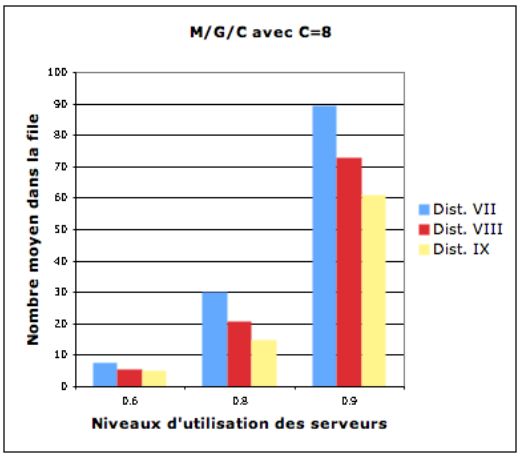
TAB. 5.5 – Les paramètres et les propriétés de distributions avec des coefficients de variation différents

Les performances d'une file multiserveur sont très dépendantes des propriétés d'ordre supérieur des distributions

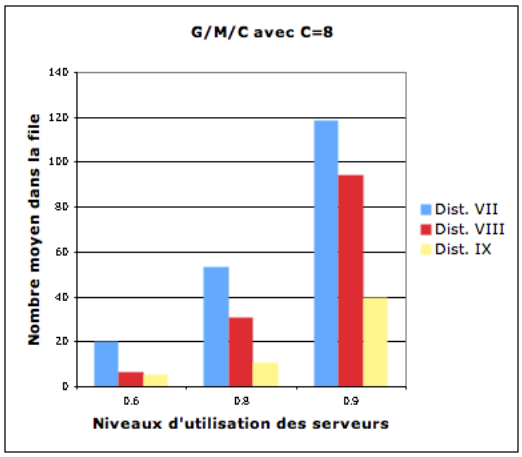
Plus généralement, les performances des files multiserveurs M/G/C, G/M/C et G/G/C, et en particulier leurs paramètres de performances moyens, peuvent être très

sensibles aux propriétés d'ordre supérieur des distributions de leur temps de service ou de leur temps d'inter-arrivées.

Pour une file M/G/C avec 8 serveurs, la Figure 5.7(a) montre la nette influence que les propriétés d'ordre supérieur à 2 de la distribution du temps de service peuvent avoir en indiquant le nombre moyen de clients dans la file pour différentes distributions du temps de service avec les deux mêmes premiers moments. Ces distributions sont décrites en détail dans la Table 5.6. En effet, le nombre moyen de clients dans la file varie sensiblement, pouvant passer pour un taux d'utilisation de 0.9 de 60 à 90, selon la distribution considérée pour le temps de service. Il est intéressant de noter que les effets apparaissent de façon significative même pour une utilisation modérée des serveurs égale à 0.6.



(a) Nombre moyen de clients dans une file M/G/C pour des coefficients de variation égaux



(b) Nombre moyen de clients dans une file G/M/C pour des coefficients de variation égaux

FIG. 5.7 – L'effet des distributions du temps de service dans les files multiserveurs M/G/C et G/M/C

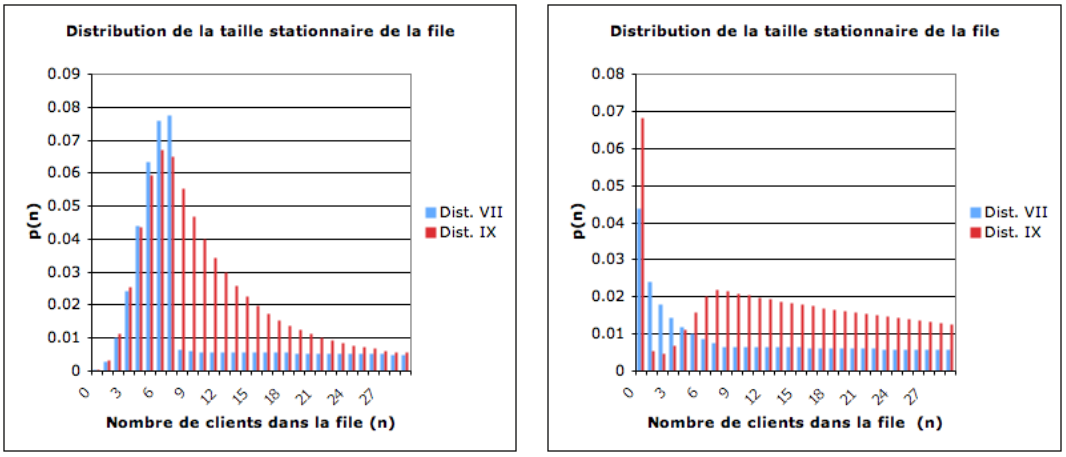
Distribution	Temps de service moyen	Coefficient de variation	Coefficient de dissymétrie	Coefficient d'aplatissement	Taux de service de l'étage 1	Probabilité d'aller à l'étage 2	Taux de service de l'étage 2
Dist. VII	1	5	7.51	7.53E01	10.00	6.98E-02	7.68E-02
Dist. VIII	1	5	2.40E01	7.85E02	1.43	2.44E-02	7.32E-03
Dist. IX	1	5	7.02E01	6.79E03	1.11	8.26E-03	8.26E-04

TAB. 5.6 – Les paramètres et les propriétés des distributions du temps de service et du temps d'inter-arrivées avec les deux mêmes premiers moments

De façon similaire, pour une file G/M/C, les propriétés d'ordre supérieur de la dis-

tribution du temps d'inter-arrivées peuvent avoir un effet très important sur le nombre moyen de clients dans la file comme le montre la Figure 5.7(b). Notons que les taux des étages des Cox-2 du processus d'arrivée ont été proportionnellement ajustés de façon à pouvoir considérer le comportement de la file à plusieurs niveaux d'utilisation de ses serveurs. Enfin, il est clair que pour une file G/G/C, comme nous l'avons vérifié, les dépendances des propriétés distributionnelles d'ordre supérieur du temps de service et du temps d'inter-arrivées peuvent être très influentes sur les paramètres de performances moyens de la file.

Comme on peut s'y attendre, les propriétés d'ordre supérieur des distributions du temps de service et du temps d'inter-arrivées n'influencent pas seulement le nombre moyen de clients dans la file, mais également la forme de la distribution stationnaire du nombre de clients dans la file. Cette dépendance est illustrée respectivement par les Figures 5.8(a) et 5.8(b) pour des files M/G/C et G/M/C avec une utilisation des serveurs de 0.9. Dans les deux cas, la file considérée dispose de 8 serveurs et le coefficient de variation de la distribution non-exponentielle est de 5. Les distributions considérées dans cet exemple sont celles des distributions Dist. VII et Dist. IX, décrites dans la Table 5.6 et qui ne diffèrent que par leurs propriétés d'ordre supérieur à 2. Nous observons sur les Figures 5.8(a) et 5.8(b) que les probabilités stationnaires d'avoir n clients dans la file à tout instant, $p(n)$, sont très différentes pour ces deux distributions.



(a) pour une file M/G/C (b) pour une file G/M/C

FIG. 5.8 – L'effet des propriétés distributionnelles d'ordre supérieur sur la distribution stationnaire du nombre de clients à tout instant dans une file avec 8 serveurs et un niveau d'utilisation de 0.9

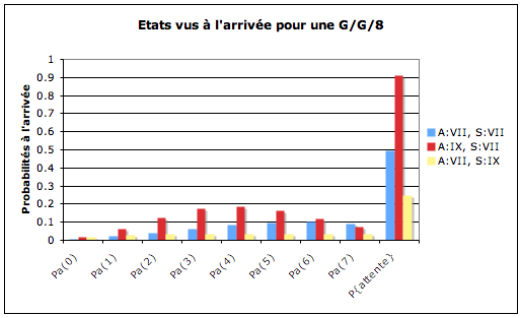


FIG. 5.9 – L'effet des propriétés distributionnelles d'ordre supérieur sur l'état stationnaire vu à l'arrivée pour une file G/G/8 avec un niveau d'utilisation de 0.5

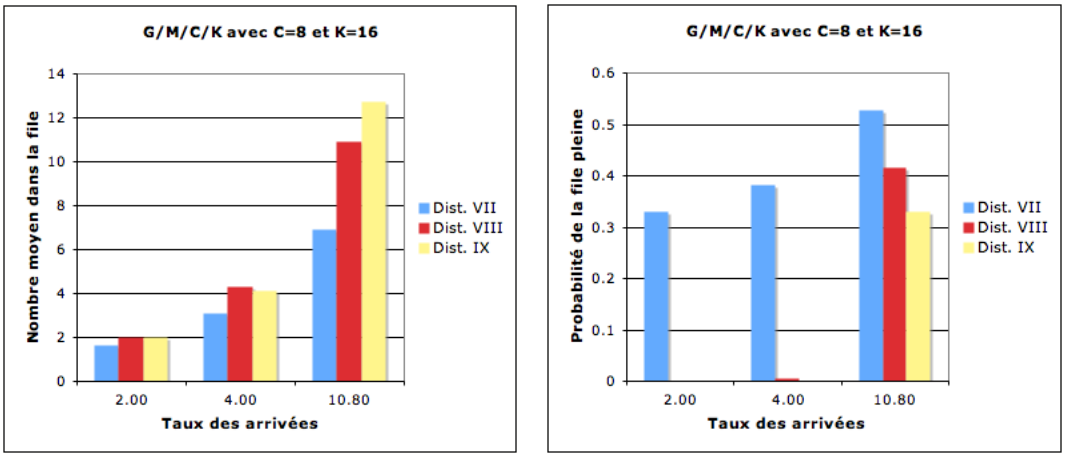
Puisque les probabilités stationnaires d'une file multiserveur dépendent des propriétés d'ordre supérieur des distributions du temps de service et du temps d'inter-arrivées, on peut s'attendre à ce qu'il en soit de même pour la probabilité de l'état trouvé par un client entrant dans la file, $p_a(n)$. La Figure 5.9 le confirme en montrant un exemple de cette dépendance pour une file G/G/C avec 8 serveurs et une utilisation des serveurs égale à 0.5. Nous avons représenté les valeurs de $p_a(n)$ pour $n = 0, 1, \ldots 7$, ainsi que la probabilité qu'un client entrant dans la file trouve tous les serveurs occupés et doive alors attendre, pour des distributions du temps de service et du temps d'inter-arrivées qui ne diffèrent à nouveau que par leurs propriétés d'ordre supérieur à 2. Ces distributions en Cox-2 sont listées dans la Table 5.6. Les légendes de cette figure identifient la distribution utilisée pour le temps d'inter-arrivées (A) et pour le temps de service (S). Par exemple, en comparant les résultats pour A :VII, S :VII avec ceux pour A :IX, S :VII, on observe les effets des propriétés d'ordre supérieur du temps d'inter-arrivées tandis que la comparaison des résultats pour A :VII, S :VII et A :VII, S :IX, illustre les effets des propriétés d'ordre supérieur du temps de service. Il est intéressant de remarquer que l'effet des propriétés distributionnelles d'ordre supérieur sur la probabilité d'attente d'un client entrant dans une file multiserveur peut être très important. Cette probabilité d'attente s'élève à près de 90% dans un cas alors qu'elle est seulement de 25% dans un autre cas.

Ces dépendances se vérifient également pour une file multiserveur à capacité d'accueil limitée

Les dépendances vis-à-vis des propriétés d'ordre supérieur du temps de service et du temps d'inter-arrivées interviennent également pour des files multiserveurs à capacité d'accueil limitée. Pour l'illustrer, nous considérons le comportement d'une file

G/M/C/K avec 8 serveurs et un tampon de taille égale à 16 pour trois distributions du temps d'interarrivées ayant les mêmes deux premiers moments mais des propriétés d'ordre supérieur différentes. Ces distributions en Cox-2 sont décrites dans la Table 5.6.

La Figure 5.10(a) montre que le nombre moyen de clients dans cette file varie fortement selon la distribution considérée pour le temps d'inter-arrivées. La Figure 5.10(b) qui représente la probabilité stationnaire que cette file soit pleine, indique que cette probabilité est très dépendante des propriétés d'ordre supérieur de la distribution du temps d'inter-arrivées. En effet, la probabilité d'avoir la file pleine est tellement faible dans certains cas qu'elle est difficilement visible sur cette figure alors qu'elle peut atteindre 33% pour une autre distribution du temps d'inter-arrivées ayant les deux mêmes premiers moments.



(a) Nombre moyen stationnaire de clients dans une file G/M/C/K pour des coefficients de variation égaux

(b) Probabilité stationnaire d'avoir la file pleine dans une file G/M/C/K pour des coefficients de variation égaux

FIG. 5.10 – L'effet des distributions du temps de service dans les files multiserveurs à capacité d'accueil limitée

Au cours de cette section, nous avons mis en évidence pour une file multiserveur la nette dépendance de la distribution du nombre de clients, mais aussi du nombre moyen dans la file, vis-à-vis des propriétés d'ordre supérieur des distributions du temps de service ou du temps d'inter-arrivées. Or à notre connaissance, la quasi-totalité des approximations existantes pour les files M/G/C, G/M/C ou G/G/C [13, 15, 45, 80, 40, 98, 73, 54, 55, 105, 106] à l'exception de [66] qui considère les trois premiers moments, prennent en compte uniquement les deux premiers moments des distributions du temps de service ou du temps d'inter-arrivées. On peut donc s'interroger sur le domaine d'ap-

plicabilité de ces approximations, et tout au moins recommander de les utiliser avec beaucoup de précautions.

Rappelons qu'en pratique, lorsque l'on souhaite modéliser par une file d'attente un système donné avec une approche constructive, les fonctions de répartition du temps de service et du temps d'inter-arrivées peuvent être connues. Des algorithmes permettent alors d'identifier ces fonctions de répartition à des distributions de remplacement de type Phase ou en loi de Cox d'aussi près que l'on veut [2, 22, 91]. Une fois cette identification effectuée, la file considérée engendre une Chaîne de Markov à Temps Continu (CMTC). Il existe de nombreuses méthodes permettant de résoudre numériquement une CMTC [97, 77] et d'obtenir ainsi les probabilités stationnaires de la file. En agissant ainsi et en supposant que l'identification a été pleinement réalisée, les dépendances des moments d'ordre supérieur (décrites précédemment pour des files monoserveurs ou multiserveurs) sont intrinsèquement prises en compte dans les distributions de remplacement et donc dans les résultats obtenus. Toutefois, la résolution de la CMTC peut s'avérer très compliquée du fait de la multiplicité des transitions et des états. Notons que cette multiplicité sera naturellement plus élevée si l'identification des fonctions de répartition du temps de service et du temps d'inter-arrivées a conduit à des distributions comportant de nombreux étages. Dans le but de réduire la complexité de la CMTC, on peut être tenté de considérer des distributions de remplacement de type Phase ou en Cox avec deux étages seulement puisque, comme nous l'avons dit précédemment, de nombreuses approximations laissent penser que les deux premiers moments suffisent à décrire les temps de service et d'inter-arrivées[1]. Or, les résultats de ce chapitre ont permis de montrer que pour certaines files monoserveurs mais également pour des files multiserveurs et des réseaux de files d'attente, une identification limitée aux deux premiers moments ne suffit pas. Une suite logique à ces travaux consisterait à savoir si étant données les distributions réelles des temps de service et des inter-arrivées dans les systèmes informatiques, les dépendances décrites dans ce chapitre vis-à-vis des moments d'ordre supérieur peuvent réellement avoir lieu sur ces systèmes. Si tel est le cas, ne pas les prendre en compte constituerait une erreur à la modélisation. Notons qu'à ce sujet, l'exemple d'un contrôleur disque présenté dans le Tableau 5.3 apporte un premier élément de réponse en considérant des exemples de distributions du temps de service plus réalistes qu'une simple Cox-2.

[1]Rappelons qu'une simple Cox-2 permet de reproduire les deux premiers moments de n'importe quelle distribution pourvu que son coefficient de variation soit supérieur à $1/\sqrt{2}$.

5.4 Conclusion

Dans ce chapitre, nous avons mis en évidence certains effets méconnus des propriétés d'ordre supérieur des distributions du temps de service et du temps d'inter-arrivées sur le comportement de plusieurs files classiques de la théorie des files d'attente comme les files M/G/1, M/G/C et G/M/C. Nous avons montré que pour les files multiserveurs ainsi que pour une file M/G/1 à capacité limitée, les performances moyennes de ces files peuvent être très sensibles aux propriétés des distributions au-delà de leur deux premier moments. De plus les résultats obtenus ont montré que pour tous les modèles considérés dans ce chapitre, y compris pour une file M/G/1 ouverte à capacité illimitée, les probabilités des états stationnaires à tout instant comme celles vues à l'arrivées dépendent étroitement des propriétés d'ordre supérieur des distributions du temps de service et du temps d'inter-arrivées. Nos résultats indiquent que les effets de ces propriétés tendent à s'accroître à mesure que le coefficient de variation des distributions et que le taux d'utilisation de la file augmentent. Toutefois, nous avons montré que même pour des taux d'utilisation et des coefficients de variation du temps de service ou du temps d'inter-arrivées modérés, leur influence peut être assez significative. Enfin, nous avons montré que ces dépendances existent et peuvent également être très fortes pour un réseau fermé, même très simple dans lequel un seul serveur est non-exponentiel.

Ces résultats indiquent qu'une caractérisation de la distribution du temps de service ou du temps d'inter-arrivées se limitant aux deux premiers moments est insuffisante pour évaluer les performances moyennes d'une file multiserveur, d'un réseau fermé très simple ou même d'une file M/G/1 à capacité d'accueil limitée. La mise en évidence de ces dépendances suggère que la plupart des approximations existantes qui s'appuie uniquement sur les deux premiers moments des distributions pour évaluer les paramètres de performances d'une file doive être employée avec précaution, du moins lorsque le coefficient de variation des distributions du temps de service ou du temps d'inter-arrivées s'élève à une certaine valeur, disons aux alentours de 1.5.

Plus généralement ces travaux ont permis de montrer qu'une caractérisation d'une loi générale par ses deux premiers moments peut être insuffisante pour de nombreuses files classiques issues de la théorie des files d'attente. Nos résultats semblent également indiquer que même les propriétés d'ordre supérieur à trois du temps de service et du temps d'inter-arrivées peuvent avoir une nette influence sur les performances moyennes du modèle. Ces observations suggèrent peut-être qu'il faut rechercher une autre façon de caractériser les lois de service ou d'inter-arrivées en utilisant d'autres grandeurs que les moments classiques. Cette recherche constitue naturellement la suite logique des

travaux présentés dans ce chapitre, et à laquelle nous n'avons pas pour le moment su répondre.

Conclusion

Le projet initial de cette thèse consistait à rechercher dans le cadre de la modélisation d'un système quels sont les aspects que l'on peut automatiser, et dans quelles mesures. Nous y avons répondu dans ce rapport de thèse en abordant d'un point de vue original le calibrage et la génération automatiques de modèles.

Concernant le calibrage d'un modèle de type file d'attente, nous avons développé une méthode générale qui permet de déterminer efficacement et automatiquement la valeur de ses paramètres indéterminés. Pour fonctionner cette méthode requiert uniquement l'existence d'un jeu de mesures provenant du système associé au modèle concerné. Nous avons également mis cette méthode de calibrage à contribution pour concevoir une nouvelle approche de modélisation, nommée HLM (« High Level Modeling »), qui s'adresse en premier lieu aux systèmes opérationnels de type « boîte noire » pour lesquels on dispose uniquement de mesures d'entrée/sortie. Cette approche de modélisation dont les conditions de fonctionnement et la démarche sont très éloignées des approches habituelles permet de délivrer simplement et rapidement un modèle immédiatement exploitable du système considéré.

Résumé des contributions

Le second chapitre de ce rapport de thèse a présenté une méthode de calibrage destinée aux modèles de type file d'attente comportant jusqu'à une dizaine de paramètres indéterminés. La recherche des valeurs de ces paramètres est basée uniquement sur la connaissance d'un jeu de mesures et nous l'avons formulée comme un problème d'optimisation continue. Pour y parvenir, nous avons proposé des définitions de la fonction objectif simples à mettre en oeuvre et permettant d'évaluer avec justesse la qualité d'un calibrage pour des modèles et des jeux de mesures de nature variée. Concernant l'algorithme de recherche, nous avons développé un algorithme de type DFO (« Deriva-

tive Free Optimization ») qui ne fait aucune hypothèse sur la fonction objectif, ni sur des propriétés du modèle à calibrer, permettant ainsi son utilisation sur de nombreux modèles. Par ailleurs, cet algorithme DFO s'accommode simplement et efficacement de contraintes non-linéaires, ce qui représente un avantage décisif puisque la plupart des problèmes de calibrage sont soumis à de telles contraintes. Au final, la méthode de calibrage proposée est très générale, simple d'utilisation, applicable automatiquement, et généralement rapide à exécuter.

Le troisième chapitre de ce rapport a été consacré à la description du fonctionnement de la méthode de modélisation HLM. Cette méthode de modélisation originale se distingue des méthodes existantes d'une part parce qu'elle repose uniquement sur des mesures, et d'autre part parce qu'elle est entièrement automatique. Dans l'optique de constituer un ensemble de modèles génériques de type file d'attente permettant de représenter un spectre assez large de comportements observés sur les systèmes informatiques, nous avons défini, en plus de modèles simples issus de la théorie des files d'attente, des modèles plus originaux permettant par exemple de reproduire des comportements qu'aucun modèle simple ne peut reproduire. Nous avons notamment présenté les modèles imbriqués, dont le fonctionnement vise à représenter de façon simple et générale le phénomène d'élongation du temps de service que l'on retrouve dans un certain nombre de systèmes informatiques. Nous avons également décrit comment utiliser la méthode de calibrage présentée dans le second chapitre afin de déterminer le calibrage de chacun de ces modèles au vue du jeu de mesures considéré. Ensembles, ces deux étapes constituent les pièces maîtresses de la méthode HLM. Le fonctionnement de cette méthode implique un certain nombre de limites comme l'existence et l'instrumentation du système considéré, l'impossibilité de garantir qu'un modèle puisse être trouvé et le fait que certains aspects des modèles produits par la méthode HLM puissent difficilement être reliés à des propriétés physiques du système. Ceci étant dit, la méthode HLM présente des avantages inhabituels et indéniables pour une méthode de modélisation. Premièrement, son exploitation ne suppose aucune connaissance intrusive sur le système, ni aucune compétence particulière en modélisation ce qui permet de rendre son utilisation accessible au plus grand nombre. Deuxièmement, la méthode HLM est entièrement automatique ce qui permet de l'embarquer dans un outil de modélisation automatique, rapide et simple à utiliser.

Enfin, et c'est l'objet du quatrième chapitre de ce rapport, la méthode HLM a permis de trouver des modèles relativement simples qui parviennent à reproduire assez finement le comportement d'entrée/sortie de systèmes multiples et variés (comme par exemple, des processeurs, des contrôleurs disques, des serveurs Web, des réseaux

Ethernet, des réseaux sans-fil, des chemins sur un réseau maillé sans fil). Par ailleurs, comme l'illustrent également les exemples considérés dans ce chapitre, les applications possibles des modèles délivrés par la méthode HLM sont nombreuses ; d'une manière générale, les modèles trouvés peuvent servir à mieux décrire, mieux comprendre ou mieux prévoir le comportement du système étudié.

Le cinquième chapitre de ce rapport, dont le contenu traite d'un autre aspect de la modélisation, a permis de montrer l'influence des propriétés distributionnelles d'ordre supérieur pour des files classiques de la théorie des files d'attente. Par exemple, bien que les performances moyennes dans une file M/G/1 ne dépendent que des deux premiers moments de la distribution du temps de service, nous avons montré que les formes de leur distribution sont très sensibles à l'ensemble de la distribution du temps de service. Concernant les files multiserveurs (comme par exemple, la file M/G/C ou G/M/C), nous avons notamment montré que la distribution du nombre de clients dans ces files, et en particulier leur valeur moyenne, dépend très fortement des propriétés distributionnelles du temps de service et du temps entre les arrivées au-delà de leurs deux premiers moments, contrairement à ce que pourrait laisser penser la plupart des approximations existantes.

Les travaux de cette thèse sur la méthode de modélisation HLM ont conduit à plusieurs publications : deux articles dans des conférences internationales et un article dans une conférence nationale [8, 9, 6]. La méthode de calibrage automatique fait actuellement l'objet d'une soumission dans une revue internationale. Enfin, les résultats obtenus sur les effets des propriétés distributionnelles d'ordre supérieur ont conduit à une première publication dans une conférence nationale [7] et d'autres articles sont actuellement en cours de soumission dans des revues internationales.

Perspectives de recherche

Parmi les travaux futurs concernant la méthode de calibrage, il serait intéressant de voir si moyennant quelques ajustements, elle pourrait être utile pour dimensionner les paramètres d'un système en cours de conception. En supposant que l'on dispose d'un modèle constructif pour un système à dimensionner et que l'on parvienne à exprimer sous forme de jeu de mesures les objectifs de performances du système, il est alors envisageable que la méthode de calibrage permette de trouver efficacement le calibrage optimal du système permettant de satisfaire les objectifs de performances. Ces travaux nécessiteraient probablement d'arranger la définition de la fonction objectif afin de prendre en compte, en plus de la proximité aux objectifs poursuivis, les coûts budgétaires associés aux paramètres.

Nous avons présenté plusieurs exemples d'applications possibles pour la méthode HLM. Toutefois, nous pensons qu'il existe d'autres applications envisageables qui mériteraient d'être explorées. En particulier, la méthode HLM pourrait être utilisée dans le cadre d'une modélisation à la volée d'un système pour lequel on a des mesures en-ligne. En effet, son exécution rapide combinée à la définition que nous avons présentée d'une fonction objectif préparée à ce type de mesures en font probablement une solution adaptée à un contexte dynamique (dans lequel les mesures pourraient être reçues de façon continue). La recherche du modèle se ferait donc en-ligne et pourrait, par exemple, servir à la détection d'anomalies. Contrairement aux approches existantes dont le mécanisme de détection repose sur l'écart statistique d'une nouvelle mesure aux mesures précédentes, l'utilisation de la méthode HLM pourrait permettre de lever une alarme uniquement lorsqu'une mesure contrarie le modèle lauréat obtenu avec les points de mesure précédents (ce qui permettrait de ne pas déclencher une alarme à chaque fois qu'une nouvelle mesure « légitime » mais éloignée des précédentes car relative à un niveau de charge du système encore jamais mesuré est obtenue). Dans cette optique, le réseau maillé sans fil MeshDVNet que nous avons présenté dans le Chapitre 4 constitue probablement une plateforme adaptée pour démarrer les premières expérimentations.

Par ailleurs, nous allons démarrer dans les prochaines semaines une collaboration avec la DIRIF (Direction Interdépartementale des Routes en Ile de France) afin de voir si la méthode HLM peut être également appliquée à des mesures provenant du trafic routier. Les applications qui pourraient déboucher des modèles obtenus sont encore imprécises et restent à définir.

Enfin, pour le dernier axe des contributions de cette thèse traitant des effets des propriétés distributionnelles d'ordre supérieur pour des files classiques, un approfondissement possible des travaux consisterait à rechercher s'il est possible d'identifier quels sont les aspects, du moins ceux les plus importants, de la distribution du temps de service qui interviennent dans l'évaluation des paramètres de performances moyens d'une file multiserveur comme la M/G/C.

Bibliographie

[1] A. Agrawala and R. Larsen. Experience with the central server model on a lightly loaded system. In *Proceedings of the 4th symposium on Simulation of Computer Systems*, pages 103–106, 1976.

[2] A.O. Allen. *Probability, statistics and queuing theory with computer science applications*, volume 2. San Diego : AcademicPress, 1990.

[3] S. Alouf, P. Nain, and D. Towlsey. Inferring network characteristics via moment-based estimators. In *Proceedings of INFOCOM*, pages 1045–1054, 2001.

[4] M. Avriel. *Nonlinear Programming : Analysis and Methods*. Dover Publishing, 2003.

[5] B. Baynat. *Théorie des files d'attente*. Hermes, 2000.

[6] T. Begin, B. Baynat, S. Fdida, F. Sourd, S. Kedad-Sidhoum, and A. Brandwajn. Génération automatique de modèles calibrés. Une méthodologie complète. In *Proceedings of Colloque Francophone sur l'Ingénierie des Protocoles - CFIP'06*, 2008.

[7] T. Begin and A. Brandwajn. Note sur les temps de service résiduels dans les systèmes type M/G/c. In *Proceedings of Colloque Francophone sur l'Ingénierie des Protocoles - CFIP'08*, 2008.

[8] T. Begin, A. Brandwajn, B. Baynat, B. Wolfinger, and S. Fdida. High-level Approach to Modeling of Observed System Behavior. In *Proceedings of SIGMETRICS Performance Evaluation Review 2007 - Performance 2007*, pages 34–36, 2007.

[9] T. Begin, A. Brandwajn, B. Baynat, B. Wolfinger, and S. Fdida. Towards an Automatic Modeling Tool for Observed System Behavior. In *Proceedings of European Performance Evaluation Workshop - EPEW 2007*, pages 200–212, 2007.

[10] D.J. Bertsimas, D. Gamarnik, and J.N. Tsitsiklis. Estimation of time-varying parameters in statistical models ; an optimization approach. In *Proceedings of the Tenth Annual Conference on Computational Learning Theory*, pages 314–324, 1997.

[11] D.J. Bertsimas and L.D. Servi. Deducing queueing from transactional data : the queue inference engine, revisited. *Operations Research*, 40(S2) :217–228, 1992.

[12] A. Björck. *Numerical Methods for Least Squares Problems*. Cambridge, 1996.

[13] M. Björklund and A. Elldin. A practical method of calculation for certain types of complex common control systens. Ericsson Technics, 1964.

[14] A.B. Bondi and W. Whitt. The influence of service-time variability in a closed network of queues. *Performance Evaluation*, 6(3) :219–234, 1986.

[15] O.J. Boxma, J.W. Cohen, and N. Huffels. Approximations of the mean waiting time in an M/G/s queueing system. *Operations Research*, 27 :1115–1127, 1979.

[16] A. Brandwajn. Equivalence and decomposition in queueing systems - a unified approach. *Performance Evaluation*, 5 :175–186, 1985.

[17] A. Brandwajn and T. Begin. A Novel Approach to G/C2/c-type queues. Technical report, UCSC, December 2007.

[18] A. Brandwajn and T. Begin. A Note on the Effects of Service Time Distribution in the M/G/1 Queue . In *Proceedings of the SPEC Benchmark Workshop 2009*, pages 138–144, 2009.

[19] A. Brandwajn and Y.L. Jow. A note on service interruptions. *SIGMETRICS Performance Evaluation Review*, 13(2) :140–148, 1985.

[20] A. Brandwajn and H. Wang. A conditional probability approach to M/G/1-like queues. *Performance Evaluation*, 65(5) :366–381, 2008.

[21] K.P. Burnham and D.R. Anderson. *Model selection and multimodel inference : a practical information-theoretic approach*. Springer-Verlag, 2nd edition, 2002.

[22] W. Bux and U. Herzog. The Phase Concept : Approximation of Measured Data and Performance Analysis. In K.M. Chandy and M. Reiser, editors, *Proceedings of the International Symposium on Computer Performance Modeling, Measurement and Evaluation*, pages 23–38. North Holland, 1977.

[23] J. Cao, M. Andersson, C. Nyberg, and M. Kihl. Web Server Performance Modeling using an M/G/1/K*PS Queue. In *Proceedings of 10th International Conference on Telecommunications - ICT'2003*, pages 1501–1506, 2003.

[24] L. Chandran-Wadia, S. Mahajan, and S. Iyer. Throughput performance of the distributed and point coordination functions of an IEEE 802.11 wireless LAN. In *Proceedings of the 15th international conference on Computer communication - ICCC '02*, pages 36–49. International Council for Computer Communication, 2002.

[25] W. Chang. Queues with Feedback for Time-Sharing Computer System Analysis. *Operations Research*, 16(3) :613–627, 1968.

[26] J.W. Cohen. *The Single Server Queue*. North-Holland Publishing Company, 1982.

[27] J.W. Cohen. *On Regenerative Processes in Queueing Theory*, volume 121 of *Lecture Notes in Economics and Mathematical Systems*. Springer Verlag, 1999.

[28] J. Cong and B. Wolfinger. A unified load generator based on formal load specification and load transformation. In *Proceedings of of ValueTools 2006 : International Conference on Performance Evaluation Methodologies and Tools*, 2006.

[29] A.R. Conn and P.L. Toint. An algorithm using quadratic interpolation for unconstrained derivative free optimization. *Nonlinear Optimization and Applications*, pages 27–47, 1996.

[30] M. Crovella, C. Lindemann, and M. Reiser. Internet performance modeling : the state of the art at the turn of the century. *Performance Evaluation*, 42(2-3) :91–108, 2000.

[31] D.J. Daley and L.D. Servi. Moment estimation of customer loss rates from transactional data. *Journal of Applied Mathematics and Stochastic Analysis*, 11(3) :301–310, 1998.

[32] D.D. Dimitrijevic. Inferring most likely queue length from transactional data. *Operations Research Letters*, 19 :191–199, October 1996.

[33] M. Drummond. *Evaluation and measurement techniques for digital computer systems*. Prentice-Hall, 1973.

[34] S. Fdida and G. Hebuterne. *Méthodes exactes d'analyse de performance des réseaux*. Hermes, 2004.

[35] S. Fdida and G. Hebuterne. *Méthodes heuristiques d'analyse de performance des réseaux*. Hermes, 2004.

[36] D. Ferrari. Workload Characterization and Selection in Computer Performance Measurement. *Computer*, pages 18–24, 1972.

[37] P. Glasserman and W. Gong. Time-Changing and Truncating K-Capacity Queues from one K to Another. *Journal of Applied Probability*, 28(3) :647–655, 1991.

[38] G.A. Gray and T.G. Kolda. Algorithm 856 : APPSPACK 4.0 Asynchronous Parallel Pattern search for derivative-free optimization. *ACM Transactions on Mathematical Software*, 32 :485–507, 2006.

[39] V. Gupta, J. Dai, M. Harchol-Balter, and B. Zwart. The effect of higher moments of job size distribution on the performance of an M/G/s queueing system. *SIGMETRICS Perform. Eval. Rev.*, 35(2) :12–14, 2007.

[40] B. Halachmi and W.R. Franta. A Diffusion Approximation to the Multi-Server Queue. *Management Science*, 24(5) :522–529, 1978.

[41] I. Halachmi, I.J.B.F. Adan, J. van der Wal, J.A.P. Heesterbeek, and P. van Beek. The design of robotic dairy barns using closed queueing networks. *European Journal of Operational Research*, 124(3) :437–446, August 2000.

[42] J.C. Heffer. Steady State Solution of the M/Ek/c (infinty, FIFO) queueing system. *Journal of the Canadian Operational Research Society*, 7 :16–30, 1969.

[43] P. Heidelberger and S. Lavenberg. Computer Performance Evaluation Methodology. *IEEE Trans. Computers*, 33(12) :1195–1220, 1984.

[44] N.J. Higham. The Matrix Computation Toolbox. http ://www.ma.man.ac.uk/∼higham/mctoolbox.

[45] P. Hokstad. Approximations for the M/G/m queue. *Operations Research*, 26(3) :510–523, 1978.

[46] J.R. Jackson. Networks of waiting line. *Operations Research*, 5 :518–521, 1957.

[47] R. Jain. *The Art of Computer Systems Performance Analysis.* John Wiley & Sons, 1991.

[48] P.H. Janssen and P.S. Heuberger. Calibration of process-oriented models. *Ecological Modelling*, 83(1-2) :55–66, 1995.

[49] J.W. Jawitz. Moments of truncated continuous univariate distributions. *Advances in Water Resources*, 27 :269–281, March 2004.

[50] F. Kappel and A.V. Kuntsevich. An implementation of Shor's r-algorithm. *Computational Optimization and Applications*, 15 :193–205, 2000.

[51] J.S. Kaufman. Approximation Methods for Networks of Queues with Priorities. *Performance Evaluation*, 4 :183–198, 1984.

[52] J. Keilson. The Ergodic Queue Length Distribution for Queueing Systems with Finite Capacity. *Journal of the Royal Statistical Society*, 28(1) :190–201, 1966.

[53] M.C. Kennedy and A. O'Hagan. Bayesian calibration of computer models. *Journal of the Royal Statistical Society : Series B (Statistical Methodology)*, 63(3) :425–464, 2001.

[54] T. Kimura. Diffusion Approximation for an M/G/m Queue. *Operations Research*, 31(2) :304–321, 1983.

[55] T. Kimura. A two-moment approximation for the mean waiting time in the GI/G/s queue. *Management Science*, 32(6) :751–763, 1986.

[56] L. Kleinrock. *Queueing Systems*, volume I : Theory. Wiley-Interscience, 1975.

[57] L. Kleinrock. *Queueing Systems*, volume II : Computer Applications. Wiley-Interscience, New York, 1976.

[58] T.G. Kolda, R.M. Lewis, and V. Torczon. Optimization by Direct Search : New Perspectives on Some Classical and Modern Methods. *Society for Industrial and Applied Mathematics : SIAM*, 45(3) :385–482, 2003.

[59] D.D. Kouvatsos and N.M. Tabet-Aouel. Product-Form Approximations for an Extended Class of General Closed Queueing Networks. In *Proceedings of the 14th IFIP WG 7.3 International Symposium on Computer Performance Modelling, Measurement and Evaluation - Performance '90*, pages 301–315, 1990.

[60] P.J. Kühn. Analysis of Complex Queueing Networks by Decomposition. In *Proceedings of 8th International Teletraffic Congress - ITC*, pages 1–8, 1976.

[61] R.C. Larson. The queue inference engine : deducing queue statistics from transactional data. *Management Sciences*, 36(5) :586–601, 1990.

[62] E. Lazowska, J. Zahorjan, G. Graham, and K. Sevcik. *Quantitative System Performance*. Prentice-Hall, 1984.

[63] E. Lazowska, J. Zahorjan, and K. Sevcik. Computer System Performance Evaluation Using Queueing Network Models. *Annual Review of Computer Science*, 1, 1986.

[64] K. Levenberg. A Method for the Solution of Certain Problems in Least Squares. *Quart. Appl. Math.*, 2 :164–168, 1944.

[65] J.D.C. Little. A Proof of the Queueing Formula L = λ W. *Operations Research*, 19 :383–387, 1961.

[66] B.N.W. Ma and J.W. Mark. Approximation of the Mean Queue Length of an M/G/c Queueing System. *Operations Research*, 43(1) :158–165, 1995.

[67] R. Marie. Calculating equilibrium probabilities for λ(n)/Ck/1/N queues. In *PERFORMANCE '80 : Proceedings of the 1980 international symposium on Computer*

performance modelling, measurement and evaluation, pages 117–125, New York, NY, USA, 1980. ACM.

[68] R.A. Marie. An approximate analytical method for general queueing networks. *IEEE Trans. Softw. Eng.*, 5(5) :530–538, 1979.

[69] H. Martens and T. Naes. *Multivariate Calibration*. John Wiley & Sons, 1989.

[70] J.O. Mayhugh and R.E. McCormick. Steady State Solution of the Queue M/Ek/r. *Management Science*, 14(11) :692–712, 1968.

[71] D. Mills. A brief history of NTP time : confessions of an Internet timekeeper. *ACM Computer Communications Review*, 33(2) :9–21, 2003.

[72] N.M. Mitrou and K. Kavidopoulos. Traffic engineering using a class of M/G/1 models. *Journal of Network and Computer Applications*, 21(4) :239–271, 1998.

[73] M. Miyazawa. Approximation of the Queue-Length Distribution of an M/GI/s Queue by the Basic Equations. *Journal of Applied Probability*, 23(2) :443–458, 1986.

[74] J.J. Moré and S.M. Wild. Benchmarking Derivative-Free Optimization Algorithms. Preprint ANL/MCS-P1471-1207, December 2007. Argonne National Laboratory.

[75] P. Nain. Queueing Systems with Service Interruptions : An Approximation Model. *Performance Evaluation*, 3 :123–129, 1983.

[76] J.A. Nelder and R. Mead. A simplex method for function minimization. *The Computer Journal*, 7 :308–313, 1964.

[77] M.F. Neuts. *Matrix-Geometric Solutions in Stochastic Models*. John Hopkins, University Press, Baltimore, 1981.

[78] J. Nocedal and S.J. Wright. *Numerical Optimization*. Springer, 2006.

[79] M.M. Noel and T.C. Jannett. A new continuous optimization algorithm based on sociological models. In *Proceedings of the American Control Conference*, pages 237–242, 2005.

[80] S.A. Nozaki and S.M. Ross. Approximations in Finite-Capacity Multi-Server Queues with Poisson Arrivals. *Journal of Applied Probability*, 15(4) :826–834, 1978.

[81] K. Olukotun, M. Heinrich, and D. Ofel. Digital System Simulation : Methodologies and Examples. In *Proceedings of 5th IEEE/ ACM Design Automation Conference*, pages 658–669, 1998.

[82] R.O. Onvural. Survey of closed queueing networks with blocking. *ACM Comput. Surv.*, 22(2) :83–121, 1990.

[83] M.J.D. Powell. Unconstrained minimization algorithms without computation of derivatives. *Bollettino della Unione Matematica Italiana*, 9 :60–69, 1974.

[84] M.J.D. Powell. The NEWUOA software for unconstrained optimization without derivatives. In *Proceedings of the 40th Workshop on Large Scale Nonlinear Optimization*, 2004.

[85] M.J.D. Powell. A view of algorithms for optimization without derivatives. In Cambridge NA Reports, Optimization Online Digest, June 2007.

[86] A. Quintero, Y. Elalamy, and S. Pierre. Performance evaluation of a broadband wireless access system subjected to heavy load. *Computer Communications*, 27(9) :781–791, 2004.

[87] S. Rosen. *Lectures on the Measurement and Evaluation of the Performance of Computing Systems.* Society for Industrial and Applied Mathematics : SIAM, New York, 1976.

[88] P.E. Le Roux. Wireless Testbed. www-rp.lip6.fr/∼leroux/meshdv/files/meshdv_plateform-1.0.pdf, 2007.

[89] R. Sadre, B.R. Haverkort, and P. Reinelt. A Fixed-Point Algorithm for Closed Queueing Networks. In *Proceedings of European Performance Evaluation Workshop - EPEW 2007*, pages 154–170, 2007.

[90] K. Salamatian and S. Fdida. A framework for interpreting measurement over Internet. In *Proceedings of the ACM SIGCOMM workshop on Models, methods and tools for reproducible network research*, pages 87–94, 2003.

[91] Y. Sasaki, H. Imai, M. Tsunoyama, and I. Ishii. Approximation of probability distribution functions by coxian distribution to evaluate multimedia systems. *Systems and Computers in Japan*, 35(2) :16–24, 2004.

[92] A. Scherr. An Analysis of Time-Shared Computer Systems. MIT Press, Cambridge, MA, 1967.

[93] D. Schiller. System capacity and performance evaluation. *j-IBM-SYS-J*, 19(1) :46–67, 1980.

[94] L.P. Seelen. An algorithm for Ph/Ph/c queues. *European Journal of Operational Research*, 23(1) :118–127, January 1986.

[95] B.W. Silverman. *Density Estimation for Statistics and Data Analysis.* Chapman & Hall/CRC, April 1986.

[96] F.J. Solis and J.B. Wets. Minimization by random search techniques. *Mathematics of Operations Research*, 6 :19–30, 1981.

[97] W.J. Stewart. *Introduction to the Numerical Solution of Markov Chains*. Princeton University Press, 1994.

[98] Y. Takahashi. Asymptotic Exponentiality of the Tail of the Waiting-Time Distribution in a PH/PH/c Queue. *Advances in Applied Probability*, 13(3) :619–630, 1981.

[99] Y. Takahashi and Y. Takami. A Numerical Method for the Steady-State Probabilities of a GI/G/s Queueing system in a General Class. *Journal of the Operations Research Society of Japan*, 19 :147–157, 1986.

[100] A. Tirumala, F. Qin, J. Dugan, J. Ferguson, and K. Gibbs. Iperf - The TCP/UDP bandwidth measurement tool. http ://dast.nlanr.net/Projects/Iperf/.

[101] E. Varki, A. Merchant, J. Xu, and X. Qiu. Issues and challenges in the performance analysis of real disk arrays. *IEEE Trans. Parallel Distrib. Syst.*, 15 :559–574, 2004.

[102] R.J. Walstra. Nonexponential networks of queues : a maximum entropy analysis. In *Proceedings of the 1985 ACM SIGMETRICS conference on Measurement and modeling of computer systems*, pages 27–37, New York, NY, USA, 1985. ACM.

[103] J. Wang and S. Keshav. Efficient and Accurate Ethernet Simulation. In *Proceedings of the 24th Annual IEEE Conference on Local Computer Networks - LCN '99*, page 182, Washington, DC, USA, 1999. IEEE Computer Society.

[104] W. Whitt. The Effect of Variability in the GI/G/s Queue. *Journal of Applied Probability*, 17(4) :1062–1071, 1980.

[105] W. Whitt. Approximations for the GI/G/m Queue. *Production and Operations Management*, 2(2) :114–161, 1993.

[106] W. Whitt. A Diffusion Approximation for the G/GI/n/m Queue. *Operations Research*, 52(6) :922–941, 2004.

[107] R.W. Wolff. The Effect of Service Time Regularity On System Performance. Computer Performance, 1977.

[108] J. Wu. Maximum Entropy Analysis of Open Queueing Networks with Group Arrivals. *The Journal of the Operational Research Society*, 43(11) :1063–1078, 1992.

Table des figures

Liste des tableaux

Modélisation et Calibrage Automatiques de Systèmes

Résumé. La pression économique qui s'exerce sur les acteurs technologiques est telle que l'évaluation de performances constitue un passage stratégique et quasi-incontournable pour les produits d'aujourd'hui. Les systèmes informatiques et les réseaux de communication modernes n'échappent pas à cette exigence. Que ce soit pour la conception, ou pour configurer et améliorer un système opérationnel, les décisions sont habituellement prises en s'appuyant sur un modèle du système. Les outils mis en oeuvre pour la modélisation du système peuvent varier selon les domaines et les besoins. Cependant, la complexité croissante, la taille grandissante et la connaissance parfois limitée des systèmes informatiques et de communications rendent cette opération difficile, si ce n'est impossible dans certains cas.

Ce rapport de thèse s'intéresse au calibrage et à la génération automatiques de modèles. Une nouvelle méthode de modélisation, nommée HLM (« High Level Modeling ») est proposée afin de permettre la modélisation rapide et automatique de systèmes opérationnels de type « boîte noire » pour lesquels on dispose uniquement de mesures. Le principe de la méthode HLM repose sur une approche originale qui consiste à rechercher parmi un ensemble présupposé de modèles génériques, si une fois correctement calibré, l'un de ces modèles permet de reproduire le comportement d'entrée/sortie du système considéré tel qu'il est décrit par les mesures. Ce rapport de thèse comprend également la description d'une méthode de calibrage automatique permettant de déterminer la valeur des paramètres indéterminés d'un modèle présupposé de type file d'attente à partir d'un ensemble de mesures. La recherche du calibrage est formulée comme un problème d'optimisation numérique pour lequel un algorithme de recherche efficace est proposé.

Le premier chapitre de ce rapport de thèse définit des notions clés à la compréhension des chapitres suivants. Le second chapitre présente le fonctionnement, la mise en oeuvre ainsi que les performances de la méthode de calibrage automatique proposée qui s'adresse aux modèles de type file d'attente. Le troisième chapitre décrit la méthode de modélisation HLM en présentant les modèles génériques considérés et la façon de les calibrer aux mesures. Le quatrième chapitre montre des exemples d'applications possibles de la méthode HLM sur des systèmes réels et variés. Enfin, le cinquième chapitre de ce rapport concerne l'influence que peuvent avoir les propriétés distributionnnelles d'ordre supérieur sur les performances de files classiques de la théorie des files d'attente.

Mots clés Modélisation, Calibrage Automatique, Boîte noire, Mesures, Files d'attente, Moments d'ordre supérieur

---