Complexité des Jeux

Edouard Bonnet [1] edouard.bonnet@dauphine.fr
Abdallah Saffidine [2] abdallahs@cse.unsw.edu.au

Les jeux exercent une fascination sur la plupart des mathématiciens et des informaticiens, ainsi que sur toute personne aimant essayer de résoudre des problèmes. Si certains auteurs appelent *jeu* tout problème de décision [11], nous essayerons dans cet article de nous limiter à des problèmes de décision comportant une dimension ludique majeure, aussi vague soit cette définition [13]. Nous distinguerons les puzzles tels le Rubik's cube ou le Sudoku et les jeux à deux joueurs (et information parfaite et complète) tels les Échecs ou les Dames.

L'objectif de cet article est de rappeler quelques définitions et résultats autour de la complexité des jeux et puzzles et de fournir des liens vers des articles plus détaillés sur le sujet. Après avoir défini de manière informelle ce que sont puzzles et jeux et ce que signifie résoudre un jeu, nous présenterons les 3 mesures de complexité des jeux les plus courantes, à savoir la complexité de l'espace d'états et de l'arbre de jeu d'une part et la complexité computationelle d'autre part. En conlusion, une table synthétique résumera les complexités de jeux et puzzles populaires.

Définitions. Informellement, un jeu est défini par une position *initiale*, un ensemble de *règles* qui fixe quels sont les coups légaux pour chacun des joueurs et des positions *terminales* associées à un *résultat* pour chaque joueur (souvent gain, perte, et éventuellement partie nulle). Dans la suite, on s'intéressera principalement à des jeux à 1 ou à 2 joueurs, à information complète et parfaite. Une *partie* est une séquence décrivant tous les coups de la position initiale à une position terminale. Une stratégie (gagnante) pour un jeu à 1 joueur, ou *puzzle*, est tout simplement une partie (gagnante). Une stratégie pour Joueur 1 dans un jeu à 2 joueurs est un objet plus complexe : c'est un arbre enraciné en la position courante, tel que les nœuds internes sont partitionnés en $N_1 \cup N_2$. N_1 correspond aux positions où le *trait* est à Joueur 1 et ont pour seul fils le coup choisi par celui-ci. N_2 correspond aux positions où le trait est à Joueur 2 (l'adversaire) et ont pour fils tous les coups légaux de Joueur 2. La stratégie est gagnante si toutes les feuilles de l'arbre sont des positions terminale gagnantes.

Une position p du jeu est gagnante s'il existe une stratégie gagnante pour Joueur 1 enracinée en p.

On peut maintenant distinguer 3 types de résolutions pour les jeux. Un jeu est *ultra-faiblement* résolu si l'on sait si la position initiale est gagnante, par exemple le jeu de Hex est ultra-faiblement résolu pour toutes les tailles, mais on ne connait pas de stratégie gagnantes pour les positions initiales de tailles suffisement grandes. Autrement dit, la preuve que la position initiale est gagnante est non-constructive. Un jeu est *faiblement* résolu si l'on connait la valeur de la position initiale et que l'on a accès à une stratégie garantissant cette valeur pour chaque joueur. Par exemple, les dames anglaises ont été faiblement résolues en 2007, et il existe un programe garanti d'obtenir la partie nulle qu'il ait les blancs ou les noirs. Un jeu est *fortement* résolu si l'on connait la valeur de toutes les positions légales du jeu. Par exemple, de nombreuses fins de parties aux échecs sont résolues fortement ; en particulier, toutes les positions à 7 pièces ou moins.

Ces définitions ne peuvent que rester informelle à cause de l'expression "avoir accès". En théorie, le raisonnement rétrograde donne la valeur de toutes les positions d'un jeu fini, mais en pratique on souhaite un accès quasi-instantané à la solution.

Complexité de l'espace d'états et de l'arbre de jeu

Dans le cadre de sa thèse, Victor Allis a proposé deux mesures de la complexité de jeux devenues classiques [1]. La complexité de l'espace d'états d'un jeu est le nombre de positions différentes possibles pour le jeu. Une borne supérieure est souvent facile à obtenir en considérant les différents types de pièces/marqueurs et leurs positions sur le plateau. Par exemple, le morpion est joué sur un plateau de 9 cases, et chacune peut être marquée de 3 manières différentes (X, O, non-marquée). La taille de l'espace d'états est inférieure à cette borne supérieure de $3^9 = 19683$ car certaines positions ainsi envisagées ne sont pas légales : le nombre de croix ne peut qu'être égal ou supérieur de 1 au nombre de ronds, et certaines configurations avec plusieurs lignes ga-

1. Lamsade, Université Paris-Dauphine, Paris, France
2. University of New South Wales, Sydney, Australie

gnantes ne sont pas possibles non plus. Une analyse plus fine de ce jeu permet d'établir que le nombre exact de positions légales différentes est 5478, sans tenir compte des symmétries [1].

Lorsqu'une analyse fine est impossible, une méthode de Monte-Carlo permet d'obtenir une approximation du nombre de positions à partir d'une borne supérieure B et d'un prédicat déterminant si une position donnée est légale ou non. On commence par générer aléatoirement un nombre de positions candidates C, puis on compte le nombre de positions légales L parmis ces positions candidates. Ce ratio appliqué à la borne supérieure donne une estimation de la taille de l'espace d'états $\simeq {}^{BL}/_{C}$.

Soit h la hauteur de l'arbre de solution le moins profond d'un jeu donné. La complexité de l'arbre de jeu de ce jeu est le nombre de feuilles d'un arbre totalement développé jusqu'à profondeur h. Cette mesure de complexité est également difficile à calculer de manière exacte, néanmoins une approximation brute est possible. Il suffit pour cela d'estimer le facteur de branchement moyen $\overline{b}$ et la longueur moyenne d'une partie $\overline{h}$ et de prendre $\overline{b}^{\overline{h}}$. Dans le cadre des puzzles, la complexité de l'arbre de jeu est obtenu en prenant la moyenne du facteur de branchement sur différentes instances et la moyenne de la longueur de la solution la plus courtes de ces différentes instances.

Complexité Computationelle

Modèles et intuition. Un jeu définit naturellement un langage : celui des positions gagnantes pour le Joueur 1. C'est la complexité de ce langage qui est le plus souvent analysée, même si on peut tout à fait se poser d'autres questions (quelle est la complexité du langage des parties gagnables en un nombre constant de coups ? en un nombre polynomial de coups ?). On préfère souvent réfléchir en terme de *problèmes* que de *langages*. Le problème associé est : étant donné le codage binaire d'une position, déterminer l'existence d'une stratégie gagnante pour Joueur 1. Quand on s'interroge sur la complexité computationnelle d'un jeu à 1 ou 2 joueurs, une question primordiale à se poser est : peut-on toujours trouver une stratégie gagnante dont les parties sont de longueur polynomiale ? Autrement dit, pour un puzzle, y a-t-il toujours une solution de longueur polynomiale ? Et pour un jeu à 2 joueurs, y a-t-il un arbre (*i.e.*, une stratégie) gagnant de profondeur polynomiale ?

En effet, on peut montrer les résultats d'appartenance consignés en Table 1, et souvent la complétude pour la classe s'obtient également. Il existe bien sûr des exceptions : le jeu de Nim n'est pas PSPACE-complet (si P $\neq$ PSPACE) mais dans P [4]. Ce jeu présente plusieurs symptômes de résolution polynomiale : acyclicité des parties, impartialité (les coups légaux d'un joueur et de son adversaire sont les mêmes), unicité de la position terminale, etc. Pour plus de détails sur ce qui rend facile ou difficile un jeu, le lecteur peut se référer à [8].

Les puzzles à solution de longueur polynomiale (comme Sudoku, Tetris [13], etc.) sont dans NP car une plus courte partie gagnante est un certificat polynomial. [3] Les puzzles à solution de longueur exponentielle (comme Sokoban ou Rush Hour [6, 12, 7]) et les jeux à 2 joueurs dont les parties sont polynomialement bornées (tels Hex, Havannah et TwixT, Amazons [15, 2, 10], ou encore le Bridge [3]) ont la même complexité : ils sont PSPACE-complets, mais c'est une petite *coïncidence* en partie dûe à NPSPACE=PSPACE [17] et PSPACE=APTIME [5]. En effet, pour les puzzles à solution non nécessairement polynomiale, on peut encoder l'état courant du puzzle sur le ruban d'une machine de Turing et deviner (puisque NPSPACE=PSPACE [17]) le premier coup d'une plus courte partie gagnante commençant en l'état courant. Au cours de l'exécution, le ruban contient seulement la position courante qui est, de fait, polynomiale en la description du problème.

Les jeux à 2 joueurs dont les parties sont polynomialement bornées sont dans PSPACE car on peut dérouler sur le ruban chaque partie (=branche) de l'arbre complet du jeu ; et ce, en espace polynomial (un nombre polynomial de positions polynomialement représentables). Certes, l'arbre contient un nombre exponentiel de parties, mais au cours de l'exécution, le ruban ne contient qu'une seule partie. Les jeux à 2 joueurs dont les parties peuvent être de longueur exponentielle (comme les Échecs [9], le Go [16]) sont dans EXPTIME car résoudre le jeu d'accessibilité sur le graphe des positions se fait en temps polynomial en la taille du graphe, c'est-à-dire exponentiel. Remarquons qu'on peut inférer la deuxième ligne de la table 1 à partir de la première ligne avec les résultats APTIME=PSPACE et APSPACE=EXPTIME [5].

Cela constitue les arguments classiques pour les résultats d'appartenance (ou de facilité). Voyons maintenant comment montrer des résultats de complétude (ou de difficulté).

3. Si, comme pour les jeux usuels, le nombre de coups légaux dans une position donnée est au plus simplement exponentiel.

Table 1 – Complexité des jeux.

longeur des parties	polynomiale	exponentielle
1 joueur	NP(-c)	PSPACE(-c)
2 joueurs	PSPACE(-c)	EXPTIME(-c)

QBF. QBF (quantified boolean formula, en anglais) est le problème de satisfaction d'une formule logique quantifiée universellement et existentiellement. Ce problème est le "premier" problème PSPACE-complet, comme SAT est le "premier" problème NP-complet. On peut voir QBF comme un jeu à 2 joueurs entre un joueur Existentiel et un joueur Universel, qui fixent à tour de rôle la valeur des variables. Par exemple, si la formule est $\psi = \exists x_1 \forall x_2 \exists x_3 \forall x_4 \ldots \phi(x_1, x_2, x_3, x_4, \ldots)$, le joueur Existentiel choisit la valeur de x_1, puis le joueur Universel choisit la valeur de x_2, et ainsi de suite. Quand toutes les variables ont été fixées, Universel choisit une clause de ϕ et Existentiel choisit un litéral de cette clause. Existentiel gagne si le litéral qu'il choisit à la fin est satisfait (et perd sinon). On constate assez facilement que ψ est vraie ssi Existentiel a une stratégie gagnante. Cela a permis de montrer que les jeux sur les graphes Géographie Généralisée et Node Kayles étaients PSPACE-complets [18].

Géographie Généralisée. *Géographie Généralisée* est un jeu à 2 joueurs sur un graphe dirigé. Un jeton est placé est initialement placé sur un sommet du graphe. A tour de rôle, les joueurs déplacent le jeton du sommet courant à un sommet successeur. Après chaque coup, le sommet duquel est parti le jeton est retiré du graphe, ainsi que tous les arcs qui lui étaient incidents. Si un joueur ne peut plus jouer (car le jeton est sur un sommet dépourvu d'arc sortant), il perd la partie. Géographie Généralisée est PSPACE-complet sur les graphes bipartis de degré 3 [18], et même sur les graphes bipartis *planaires* de degré 3 [14]. Ce dernier résultat est très pratique [10, 2] car il permet de montrer la PSPACE-difficulté d'un jeu avec des gadgets de (i, j)-sommets pour les couples d'entier $(i, j) \in \{(0, 2), (1, 1), (1, 2), (2, 1)\}$ où un (i, j)-sommet est un sommet de degré entrant i et de degré sortant j. Le $(0, 2)$-sommet est le sommet initial. Les $(1, 2)$ et $(2, 1)$-sommets sont les autres sommets du graphe. Et deux $(1, 1)$-sommets mis bout à bout permettent d'encoder une arête du graphe.

Constraint Logic. Les deux derniers paragraphes étaient en fait consacrés à la PSPACE-complétude des jeux à 2 joueurs dont les parties sont de taille polynomiale. Quid des autres jeux et des autres classes de complexité ? Les auteurs du livre [11] présentent un framework assez complet pour dériver des résulats de difficulté allant de P-difficulté à l'indécidabilité à partir de jeu sur des graphes où le(s) joueur(s) retourne(nt) des arcs sous certaines contraintes de flot entrant, avec comme but d'arriver à retourner une arête en particulier. Ce framework est puissant mais, curieusement, n'a pas encore été utilisé par d'autres que leurs auteurs.

Conclusion

La Table 2 résume les complexités d'états, d'arbre de jeu, et computationelles de quelques jeux et puzzle populaires. [4] La référence originale de la plupart des résultats de la Table 2 peut se trouver dans [12] ou [19].

La complexité computationelle s'intéresse aux versions généralisées des jeux. Certains jeux peuvent facilement être généralisés à des tailles arbitrairement grandes. Par exemple, si l'on peut concevoir le jeu de Go sur un plateau $n \times n$ pour tout entier n, la généralisation d'Awalé ou du jeu de Backgammon est plus sujette à caution.

À l'inverse, les complexités de l'espace d'états et de l'arbre de jeu, se concentrent sur une taille de jeu particulière. Par conséquent, on obtient des complexités différentes pour le jeu de Go sur plateau 9×9 et 19×19.

Références

[1] L. V. Allis. *Searching for Solutions in Games an Artificial Intelligence*. PhD thesis, Vrije Universitat Amsterdam, 1994.

[2] É. Bonnet, F. Jamain, and A. Saffidine. Havannah and Twixt are PSPACE-complete. In *8th International Conference on Computers and Games (CG)*. August 2013.

[3] É. Bonnet, F. Jamain, and A. Saffidine. On the complexity of trick-taking card games. In

4. Une liste plus exhaustive se trouve sur `http://en.wikipedia.org/wiki/Game_complexity`.

TABLE 2 – Différentes Complexités pour quelques jeux et puzzles populaires. Les complexités de l'espace d'états et de l'arbre de jeu sont exprimées en logarithme à base 10.

Nombre de joueurs	Longueur des parties	Jeu ou puzzle (taille standard)	Complexité Espace d'états	Arbre de jeu	Computationelle
1 joueur		Rubik's cube (3×3×3)	19	20	?
	Polynomiale	Taquin (5×5)	25	>37	NP-complet
		SameGame (15×15)	159	85	
	Exponentielle	Sokoban (20×20)	98	280	PSPACE-complet
2 joueurs	Polynomiale	Tic-tac-toe (3×3)	3	5	PSPACE-complet
		Othello (8×8)	28	58	
		Hex (11×11)	57	98	
		Gomoku (15×15)	105	70	
		Havannah (10×10×10)	127	157	
		Amazons (10×10)	40	212	
		Bridge (4×13)	<17	<40	
	Exponentielle	Checkers (8×8)	18	31	EXPTIME-complet
		Dames (10×10)	30	54	
		Échecs (8×8)	47	123	
		Shogi (9×9)	71	226	
		Go (19×19)	171	360	

23rd International Joint Conference on Artificial Intelligence (IJCAI), 2013.

[4] C. L. Bouton. Nim, a game with a complete mathematical theory. *Annals of Mathematics*, 3(1) :35–39, 1901.

[5] A. K. Chandra, D. C. Kozen, and L. J. Stockmeyer. Alternation. *Journal of the ACM*, 28(1) :114–133, 1981.

[6] J. C. Culberson. Sokoban is PSPACE-complete. In *Proceedings in Informatics*, volume 4, pages 65–76, 1999.

[7] G. W. Flake and E. B. Baum. *Rush Hour* is PSPACE-complete or "why you should generously tip parking lot attendants". *Theoretical Computer Science*, 270(1) :895–911, 2002.

[8] A. S. Fraenkel. Nim is easy, chess is hard — but why ? ? *ICGA Journal*, 29(4) :203–206, 2006.

[9] A. S. Fraenkel and D. Lichtenstein. Computing a perfect strategy for $n \times n$ Chess requires time exponential in n. *Journal of Combinatorial Theory, Series A*, 31(2) :199–214, 1981.

[10] T. Furtak, M. Kiyomi, T. Uno, and M. Buro. Generalized Amazons is PSPACE-complete. In *19th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 132–137, 2005.

[11] R. A. Hearn and E. D. Demaine. *Games, Puzzles, and Computation*. A K Peters, July 2009.

[12] A. Junghanns and J. Schaeffer. Sokoban : Enhancing general single-agent search methods using domain knowledge. *Artificial Intelligence*, 129(1) :219–251, 2001.

[13] G. Kendall, A. J. Parkes, and K. Spoerer. A survey of NP-complete puzzles. *ICGA Journal*, 31(1) :13–34, 2008.

[14] D. Lichtenstein and M. Sipser. Go is polynomial-space hard. *Journal of the ACM*, 27(2) :393–401, 1980.

[15] S. Reisch. Hex ist PSPACE-vollständig. *Acta Informatica*, 15(2) :167–191, 1981.

[16] J. M. Robson. The complexity of Go. In *IFIP*, pages 413–417, 1983.

[17] W. J. Savitch. Relationships between nondeterministic and deterministic tape complexities. *Journal of Computer and System Sciences*, 4(2) :177–192, 1970.

[18] T. J. Schaefer. On the complexity of some two-person perfect-information games. *Journal of Computer and System Sciences*, 16(2) :185–225, 1978.

[19] H. J. van den Herik, J. W.H.M. Uiterwijk, and J. van Rijswijck. Games solved : Now and in the future. *Artificial Intelligence*, 134(1) :277–311, 2002.