31593 CARREZ Valentin

Résolution de niveaux du Sokoban

Depuis que j'ai découvert le Sokoban, un jeu de réflexion japonais, dans un tutoriel de programmation, il me fascine par sa difficulté malgré ses règles simples. Mon binôme connaissant aussi ce jeu, nous avons pensé que ce TIPE serait une bonne occasion de travailler sur ce sujet d'intérêt commun.

Le Sokoban, littéralement « gardien d'entrepôt », consiste à pousser des caisses sur des objectifs dans un labyrinthe. On simule ainsi la gestion d'un entrepôt ou un problème de livraison de colis (en assimilant le labyrinthe à une ville), deux enjeux importants dans les villes où les livraisons sont nombreuses.

Ce TIPE fait l'objet d'un travail de groupe.

Liste des membres du groupe :

- *Anonyme*

Positionnement thématique (ÉTAPE 1) :

- *INFORMATIQUE (Informatique pratique)*
- *INFORMATIQUE (Informatique Théorique)*

Mots-clés (ÉTAPE 1) :

Mots-clés (en français)	Mots-clés (en anglais)
Jeu vidéo de réflexion	*Puzzle video game*
Problème de recherche	*Search problem*
PSPACE-complet	*PSPACE-complete*
Émondage de graphe	*Graph pruning*
Heuristique	*Heuristic*

Bibliographie commentée

Le Sokoban est un jeu vidéo inventé en 1981 par Hiroyuki Imabayashi. Il consiste à déplacer des caisses vers des cibles dans un labyrinthe en deux dimensions. Le joueur est astreint à se déplacer selon les quatre directions (nord, sud, est, ouest). Il ne peut bouger qu'une seule caisse à la fois et ne peut pas les tirer. Ces règles simples en font un jeu populaire, bien que difficile [2].

En effet, il a été démontré que la résolution d'un Sokoban est un problème NP-difficile et PSPACE-complet [4]. Timo Virkkala [2] explique que la difficulté du Sokoban provient à la fois de la très grande profondeur et du facteur de branchement (qui peut aller jusqu'à 100) de l'arbre des possibilités. Ainsi, on se concentre plutôt sur la recherche d'une solution et non sur son optimalité, ce qui en fait un problème de recherche. La notion de solution optimale est d'ailleurs ambiguë pour le Sokoban, selon qu'on parle du nombre de déplacements du joueur ou

du nombre de poussées de caisses.

En 1999, Andreas Junghanns présente *Rolling Stone*, le premier solveur de Sokoban, dans un article qui introduit les bases des concepts de résolution. On y trouve, entre autres : la détection de configurations qui rendent le jeu insoluble (*deadlocks*), l'utilisation d'une table de transposition pour ne pas traiter plusieurs fois la même configuration, la mise en place de mouvements à grande échelle (*macro moves*) pour éviter de générer des états intermédiaires inutiles et le calcul d'une heuristique - qui se base sur la distance entre les caisses et les objectifs - pour guider sa recherche. C'est aussi dans cet article qu'est introduit l'ensemble de 90 niveaux qui sert depuis de référence pour tester les performances des solveurs [1].

Les solveurs postérieurs à *Rolling Stone* ont introduit différentes approches.
Par exemple, *Powerplan* voit le Sokoban comme un graphe abstrait de tunnels et de salles. Cette approche n'est cependant pas vraiment meilleure que celle de *Rolling Stone*, puisque *Powerplan* ne résout que 10 niveaux là où *Rolling Stone* en résolvait 54 [6].
Timo Virkkala formalise la notion d'enclos (*corral*), qui désigne une zone du niveau inaccessible au joueur à cause de caisses qui en obstruent les accès. Il examine aussi plusieurs moyens de détecter les *deadlocks*, comme par exemple le calcul de positions mortes (*dead positions*), des cases dont l'emplacement fait que le niveau devient insoluble si une caisse est poussée dessus [2].
Un autre solveur, *GroupEffort*, utilise simultanément différents algorithmes de recherche qui travaillent de manière indépendante (pas de synchronisation), les seules informations partagées étant les configurations insolubles [4].
Le premier solveur capable de résoudre l'ensemble des 90 niveaux, appelé *FESS* (pour *FEeature Search Space*), est présenté en 2020 par Yaron Shoham et Jonathan Schaeffer. Là où la plupart des autres solveurs n'utilisent qu'une seule heuristique dans leur recherche, *FESS* se distingue par son utilisation combinée de plusieurs heuristiques différentes. Il crée ainsi un espace abstrait, nommé *Feature Space*, qu'il explore conjointement avec l'arbre des possibilités du Sokoban pour trouver une solution [7].

En plus des articles présentant des solveurs, on peut trouver différentes stratégies aidant à la résolution dans la littérature. Albert L. Zobrist propose une fonction de hachage efficace pour pouvoir stocker les configurations de jeu déjà visitées lors de la recherche [5]. Le site Web *sokobano.de* introduit d'autres types de deadlocks (*freeze deadlock*, *bipartite deadlock etc.*) et propose des méthodes de détection. Il contient aussi de nombreuses statistiques sur les différents solveurs (issus de la littérature ou non).

Bien que de nombreuses idées soient présentées, peu de ressources indiquent comment les mettre en œuvre de façon concrète. Implémenter ces algorithmes de manière efficace sera donc un des enjeux de ce TIPE.

Problématique retenue

Quelles stratégies adopter pour trouver une solution le plus rapidement possible à un niveau de Sokoban ?

Objectifs du TIPE du candidat

1. Concevoir un algorithme généraliste pouvant résoudre, en théorie, n'importe quel niveau de Sokoban, sans chercher une solution optimale.
2. Analyser les niveaux afin d'identifier des motifs récurrents (*deadlocks*, tunnels *etc.*) pour élaborer des stratégies visant à accélérer la recherche.
3. Implémenter cet algorithme et ces stratégies en Java.
4. Effectuer des tests de performances sur des ensembles de niveaux pour comparer notre solveur à ceux qui existent déjà.

Références bibliographiques (ÉTAPE 1)

[1] ANDREAS JUNGHANNS : Pushing the Limits: New Developments in Single-Agent Search *: Thesis (1999), University of Alberta, https://www.researchgate.net/publication /2305703_Pushing_the_Limits_New_Developments_in_Single-Agent_Search*

[2] TIMO VIRKKALA : Solving Sokoban *: Master's Thesis (2011), University of Helsinki, http://sokoban.dk/wp-content/uploads/2016/02/Timo-Virkkala-Solving-Sokoban-Masters-Thesis. pdf*

[3] : sokobano.de *: http://sokobano.de/wiki/index.php?title=Main_Page, consulté régulièrement entre 2022 et 2023*

[4] NILS FROLEYKS : Using an Algorithm Portfolio to Solve Sokoban *: Bachelor Thesis (2016), Karlsruhe Institute of Technology, https://baldur.iti.kit.edu/theses/SokobanPortfolio.pdf*

[5] ALBERT L. ZOBRIST : A New Hashing Method with Application for Game Playing *: Technical Report (1970), University of Wisconsin, https://minds.wisconsin.edu/bitstream /handle/1793/57624/TR88.pdf*

[6] A. BOTEA, M. MÜLLER AND J. SCHAEFFER : Using abstraction for planning in Sokoban *: Proceedings of the 3rd International Conference on Computers and Games (2002), Springer, https://webdocs.cs.ualberta.ca/~mmueller/ps/botea-sokoban.pdf*

[7] YARON SHOHAM AND JONATHAN SCHAEFFER : The FESS Algorithm: A Feature Based Approach to Single-Agent Search *: IEEE Conference on Games (2020), https://ieee-cog. org/2020/papers/paper_44.pdf*

DOT

[1] *: Mai 2022 : Recherche du sujet et formulation de la problématique : décision de se restreindre à l'existence d'une solution (et non à la recherche d'une solution optimale). Familiarisation avec les recherches déjà effectuées dans ce domaine.*

[2] *: Juin 2022 : Première version du solveur : recherche par force brute sans optimisation via un parcours en profondeur de l'arbre des états. Utilisation du hash de Zobrist [5] pour éviter les cycles. Cette approche permet de résoudre les niveaux simples, et suffit étonnamment pour certains niveaux de difficulté moyenne.*

[3] *: Juillet - août 2022 : Début de la recherche de « deadlock » : ajout de la détection des « dead tiles » et des « freeze deadlocks ». Même si ces optimisations n'augmentent pas beaucoup le nombre de niveaux résolus, elles contribuent tout de même à la réduction de l'espace de recherche.*

[4] *: Septembre - octobre 2022 : Détection de tunnels et de salles. Calcul d'un ordre de rangement d'une salle dans un cas simple : les salles comportant seulement une entrée et possédant au moins une cible.*

[5] *: Décembre à janvier 2023 : Mise en place d'une recherche guidée à l'aide d'une heuristique, à la manière de A*. Implémentation de deux heuristiques, simple et gloutonne. Optimisation de la deuxième pour réduire sa complexité.*

[6] *: Février à mars 2023 : Ajout de la détection de « PI-Corral deadlocks » [3]. Cependant, les résultats ne sont pas satisfaisants, d'où la décision de considérer les I-Corrals plutôt que les PI-Corrals.*

[7] *: Mars 2023 : Implémentation de l'algorithme FESS [7], très partielle, mais suffisante pour résoudre des niveaux qui paraissaient jusque là hors de portée.*

[8] *: Avril - début juin 2023 : Lancement du solveur sur les deux ensembles de niveaux principaux. Comparaison des résultats avec les autres solveurs existants. Réalisation de la présentation.*