Sujet de TER

UNIVERSITÉ DE NANTES

Fonctions objectives pour la sélection de cibles dans StarCraft

1 Contexte

Les jeux de stratégie en temps réel peuvent être vus comme des simplifications pratiques du monde réel (règles finies, paramètres limités). Le principe est simple : gérer le développement économique et militaire d'une base afin de conquérir l'espace de l'adversaire. Le complexité de ces jeux vient du fait que le terrain (bien plus grand qu'un échiquier ou un plateau de Go) est plongé dans un brouillard de tel sorte que la visibilité de tout ce qui est autour des pièces est très limitée. Il s'agit bien ici d'un environnement dont l'état change perpétuellement, en temps réel, et où l'information est incomplète puisque nous ne savons pas en permanence ce que fait l'adversaire. Pour donner une image, cela reviendrait à faire une partie d'échec sur un échiquier 2048×2048 où les deux joueurs jouent simultanément, et où les pièces adverses sont cachées par un brouillard uniquement dissipé si l'une de nos pièces se trouve à proximité. Pour les joueurs humains, il faut également ajouter une dimension de dextérité inhérente au caractère temps réel du jeu. Pour garder notre analogie avec la partie d'échec, c'est comme si les joueurs devaient déplacer leurs pièces en jouant du piano. Nous invitons le lecteur à lire les surveys [6, 9] pour plus de détails sur l'état de l'art en intelligence artificielle pour les jeux de stratégie en temps réel.

GHOST [7] est un framework contenant un solveur de problèmes de satisfaction / optimisation de contraintes (CSP / COP) orientés pour les problèmes liés aux jeux de stratégie en temps réel tel que StarCraft : Brood War™, ainsi qu'un ensemble de problèmes pré-encodés.

Le solveur dans GHOST est une meta-heuristique de recherche locale basée sur l'algorithme Adaptive Search [4]. Le choix s'est porté sur une méta-heuristique plutôt qu'un solveur complet du fait de la très réactivité que doit avoir le solveur pour retourner une solution. Dans un jeu de stratégie en temps réel, un solveur ou un quelconque algorithme n'a qu'entre 30 et 40 millisecondes maximum pour effectuer son calcul.

Cette framework, disponible sous forme de bibliothèque C++, contient ainsi plusieurs problèmes pré-encodés que l'utilisateur lambda peut utiliser dans son programme d'intelligence artificielle (appelé "bot") pour StarCraft, tel que le problème de Build Order [1, 5] (ordonnancement d'actions afin de minimiser le temps d'achèvement d'un objectif), le problème de Wall-in [8, 10] (placement de bâtiments de manière à créer un goulot d'étranglement facilitant la défense d'un point stratégique) ou le problème de sélection de cibles [3, 2] lors d'une rixe entre deux groupes armés (maximisant certains objectifs, comme les dégâts par exemple).

2 Sujet de TER

Ce sujet de TER convient à un groupe de deux étudiants **maximum**.

La modélisation et l'implémentation actuelle du problème de sélection de cibles sont loin d'être satisfaisantes. Le but de ce TER se décompose en trois étapes :

- Concevoir de meilleures fonctions objectives pour prendre en compte les deux problèmes d'optimisation actuels, à savoir maximiser les dégâts chez l'adversaire (non trivial car différents paramètres rendent ce problème fortement combinatoire) et maximiser les unités éliminées. De nouvelles fonctions objectives pourront être proposées, comme par exemple minimiser l'*overkill* (n unités qui visent et éliminent une cible alors que $m < n$ unités auraient suffit).
- Implémenter ces fonctions objectives en C++ dans GHOST .
- Comparer les résultats obtenus avec l'implémentation actuelle. Selon les résultats, une intégration dans la framework est envisageable.

Références

[1] David Churchill and Michael Buro. Build order optimization in starcraft. In *AIIDE*, pages 14–19. The AAAI Press, 2011.

[2] David Churchill and Michael Buro. Incorporating search algorithms into RTS game agents. In *AIIDE First Workshop on Artificial Intelligence in Adversarial Real-Time Games*, 2012.

[3] David Churchill, Abdallah Saffidine, and Michael Buro. Fast heuristic search for RTS game combat scenarios. In *AIIDE*. The AAAI Press, 2012.

[4] Philippe Codognet and Daniel Diaz. Yet another local search method for constraint solving. In *proceedings of SAGA'01*, pages 73–90. Springer Verlag, 2001.

[5] Matthias Kuchem, Mike Preuss, and Günter Rudolph. Multi-objective assessment of pre-optimized build orders exemplified for starcraft 2. In *CIG*, pages 1–8. IEEE, 2013.

[6] Santiago Ontañón, Gabriel Synnaeve, Alberto Uriarte, Florian Richoux, David Churchill, and Mike Preuss. A survey of real-time strategy game AI research and competition in starcraft. *IEEE Transactions on Computational Intelligence and AI in games*, 5(4) :1–19, 2013.

[7] Florian Richoux, Jean-François Baffier, and Alberto Uriarte. GHOST : A combinatorial optimization solver for RTS-related problems. *IEEE Transactions on Computational Intelligence and AI in games*, To appear.

[8] Florian Richoux, Alberto Uriarte, and Santiago Ontañón. Walling in strategy games via constraint optimization. In *AIIDE*. The AAAI Press, 2014.

[9] Glen Robertson and Ian Watson. A review of real-time strategy game AI. *AI Magazine*, 2014.

[10] Michal Čertický. Implementing a wall-in building placement in starcraft with declarative programming. *arXiv*, 2013.