Étude sur l'application de joueurs artificiels sur un jeu de stratégie en temps réel

Dimitri Cocheril-Crèvecœur

2023-2024

Motivations

J'ai en 2020 entrepris de recoder en C++ le jeu de grande stratégie *Stellaris* en 24 bit pour une architecture d'ez80. J'ai alors abandonné devant le problème de l'intégration de joueurs artificiels. Mon sujet de TIPE aborde l'intégration de modèles de combat, et plus généralement de joueurs artificiels, dans un jeu plus documenté, *StarCraft*.

Positionnement thématique

INFORMATIQUE (informatique pratique et théorique)

Mots clés

Mots clés (Français)	**Mots clés** (Anglais)
Recherche Arborescente Monte-Carlo	Monte Carlo Tree Search (MCTS)
Stratégie en temps réel	Real-time strategy (RTS)
Joueur artificiel	Bot/agent
Modèle de combat	Combat model

Bibliographie commentée

StarCraft, un jeu de stratégie en temps réel (RTS), sous-genre des jeux de stratégie, est depuis plusieurs années source d'avancées dans le domaine de l'IA à cause de son fonctionnement complexe. [1] Une partie peut contenir entre 50 et 400 unités sur une carte de taille 128×128. [4] Ces unitées peuvent faire un certain ensemble d'actions avec une certaine latence 24 fois par secondes.

Les meilleurs bots actuels sont faits avec des modèles IA neuronales. Le but de ce TIPE est d'étudier la conception d'algorithmes de joueurs artificiels plus classiques.

J'ai ainsi mis en place un joueur utilisant l'algorithme Monte-Carlo Tree Search. Étant particuliérement gourmand en ressources, et son éxécution étant limitée dans le temps, un certain travail d'optimisation a été nécessaire ainsi que l'implémentation d'un multithreading. Le choix d'une fonction d'évaluation est également un challenge fréquent.[3][5]

Les algorithmes classiques les plus performants ont fait le choix de monter en abstraction. Si le temps me le permet, je monterais en abstraction, en groupant les unités en armées grâce à l'algorithme DBSCAN. [2][6]

Problématique retenue

Quel algorithme est le plus efficace en stratégie de combat sur un RTS ?

Objectifs du TIPE du candidat

Pour réaliser le TIPE, il a d'abord fallu développer un moteur de jeu typé *StarCraft* et une api. J'entreprends ensuite la création de joueurs artificiels à différents niveaux d'abstraction : un joueur artificiel utilisant l'algorithme MCTS multithreadé, et un joueur artificiel utilisant une technique de recherche par portfolio. Nous nous intéressons particulièrement aux phases de combat, et n'abordons l'aspect gestion de ressources que marginalement, en seconde partie d'étude.

Abstract

Real-time strategy games are challenging for the creation of bots due to their real-time and large-scale aspects. A large community of players and researchers is actively trying to create the perfect bot through different techniques. I study here two scripts that have already proven to be good enough to beat some human players, but not the best world champions.

Références bibliographiques

[1] David Churchill, Mike Preuss, Florian Richoux, Gabriel Synnaeve, Alberto Uriarte, Santiago Ontañón, and Michal Certicky. StarCraft Bots and Competitions. In Newton Lee, editor, *Encyclopedia of Computer Graphics and Games*, pages 1–18. Springer International Publishing, 2016.

[2] Michael Hahsler, Matthew Piekenbrock, and Derek Doran. dbscan : Fast density-based clustering with r. *Journal of Statistical Software*, 91(1) :1–30, 2019.

[3] Santiago Ontañón. Informed monte carlo tree search for real-time strategy games. *2016 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 1–8, 2016.

[4] 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 :293–311, 2013.

[5] Dennis J. N. J. Soemers. Tactical planning using mcts in the game of starcraft, 2014.

[6] Alberto Uriarte and Santiago Ontañón. Combat models for rts games. *IEEE Transactions on Games*, 10 :29–41, 2016.

DOT

[1] avril 2023 : Recherches sur les études déjà réalisées et les codes déjà produits en lien avec le sujet.

[2] mai/juin 2023 : Codage du moteur du jeu en C++, inspiré de la réelle API de *StarCraft*.

[3] juillet 2023 : Gros travail d'optimisation sur le moteur, principalement du pathfinding.

[4] septembre/octobre 2023 : Création et tests du MCTS, puis son perfectionnement, avec multithreading du moteur.