FORAGE DE DONNÉES

BIN 701

Clustering de plans pour des strategies de Starcraft

Travail présenté à

M. Shengrui Wang

Par

Julien Filion 07 177 770

Université de Sherbrooke

Été 2010

Table des matières

Introduction

À l'été 2010, dans le cadre du cours Forage de données (BIN701) j'ai travaillé sur un projet qui exploite l'aspect séquentiel de plan stratégique pour appliquer des concepts empruntés aux problèmes de séquençage en bio-informatique. J'ai concentré mes efforts sur la génération d'une librairie de plan stratégique à partir d'un ensemble de scénarios réel. Pour ce faire, j'ai utilisé une méthode de clustering hiérarchique sur des séquences d'actions. Toutes les démarches ont été appliquées sur Starcraft, un jeu de stratégie en temps réel.

Starcraft

Puisque Starcraft est assez complexe, je ne ferais qu'une petite introduction au jeu. Pour plus d'information sur celui-ci, je conseille la page Wikipédia[1] et le site officiel[2] de Starcraft.

Starcraft est un jeu de stratégie en temps réel qui se déroule dans un monde futuriste. Il a été lancé par Blizzard en 1998, mais demeure encore très populaire 12 ans après sa sortie. Il est particulièrement populaire en Corée du Sud où les joueurs professionnels gagnent des commandites et participent à des compétitions télévisées.

Pour ce projet, on ne s'intéresse qu'à un type de partie où deux joueurs s'affrontent en duel.

Déroulement général d'une partie

Lors de ces duels, chaque joueur commence avec une base et quelques travailleurs, et il doit détruire la base de son adversaire pour gagner la partie. Le joueur devra gérer plusieurs aspects lors d'une partie.
Le premier aspect est la récolte de ressource, le joueur devra récolter deux ressources, du minerai et du gaz, afin de pouvoir agrandir sa base, agrandir son armée, améliorer ses technologies.
Un deuxième aspect est la gestion de sa base, le joueur devra construire des bâtiments qui lui permettront entre autres de recruter des unités et d'accéder à de nouvelles technologies.
Un troisième aspect est la création de nouvelles unités, le joueur doit composer son armée afin d'avoir un avantage sur son ennemi.
Finalement, le dernier aspect est la gestion des combats, où le joueur doit attaquer son adversaire pour le détruire, mais il doit aussi protéger sa base des attaques.

[1] http://en.wikipedia.org/wiki/StarCraft
[2] http://us.blizzard.com/en-us/games/sc/

Races

Dans un duel de Stacraft, les joueurs ont le choix entre 3 races. Chaque race est complètement différente, elles possèdent leurs propres unités, bâtiments et technologies. Les Terran sont des humains exilés de la terre, ils s'adaptent facilement à toutes les situations. Les Zerg sont une race extra-terrestre qui se base sur les mutations pour prendre avantage sur l'ennemi. Les Protoss sont des extra-terrestres humanoïdes qui puisent leurs forces de leurs technologies avancées.

Stratégies

Bien que la description suivante soit spécifique de Starcraft, elle peut facilement être généralisée pour la plupart des jeux de stratégie en temps réel.
Starcraft est principalement un jeu de stratégie, on doit organiser notre armée pour avoir un avantage sur l'adversaire. Les décisions qui doivent être prises dans une partie peuvent être divisées en deux catégories, soit la microgestion et la macrogestion.

Microgestion

La microgestion (ou simplement micro) représente le contrôle des unités. Elle inclut toutes les tactiques pour attaquer l'ennemi, défendre notre base et faire de la reconnaissance.

Macrogestion

La macrogestion (ou simplement macro) représente l'utilisation des ressources. Les actions qui entrent dans cette catégorie sont principalement : la construction de bâtiment, le recrutement d'unité et l'amélioration des technologies.

Build order

Les joueurs expérimentés apprennent rapidement des séquences d'actions optimales pour la macrogestion en début de partie. Les débuts de parties sont propices à ce genre d'optimisation, car on connaît l'état initial de l'adversaire et il ne peut pas nous attaquer avant qu'il ait assemblé une armée, ce qui laisse un peu de liberté aux joueurs. Ces séquences sont appelées des builds orders. Certains build order sont popularisés et largement utilisés bien que chaque joueur puisse utiliser un build order personnalisé.

Voici un exemple de build order agressif de Zerg très court, mais assez fréquent nommé « 5 Pool » :

1. Recruter un Drone
2. Construire un Spawning Pool
3. Recruter deux Drones
4. Recruter six Zerglings
5. Recruter un Drone
6. Recruter un Overlord

Voici un deuxième exemple de build order de Zerg nommé « 12 Hatch » :

1. Recruter quatre Drones
2. Recruter un Overlord
3. Recruter trois Drones
4. Construire une Hatchery
5. Construire un Spawning Pool

Reconnaissance de plan

Au laboratoire du Planiart, un de nos projets est de faire un joueur artificiel pour le jeu de Starcraft. Un avantage important dans un tel jeu est de pouvoir prédire quelles sont les actions de l'adversaire. En connaissant la stratégie de l'adversaire, on peut adapter notre stratégie en conséquence. C'est pourquoi on a utilisé des algorithmes de reconnaissance de plans. Par contre la plupart ces algorithmes demandes une librairie de plan pour fonctionner. C'est pour cette raison que j'ai décidé de générer automatiquement des librairies de plan en analysant une grande quantité de matchs enregistrés. Pour l'instant, on ne s'intéresse qu'aux actions de macrogestion en début de partie, les builds orders.

Algorithme

Acquisition des séquences d'action

La première étape du projet est de générer des séquences d'action à partir de match enregistré. Chaque match est un duel entre deux joueurs, on peut donc extraire deux plans par match. J'ai donc fait un petit programme qui extrait la séquence d'actions de chaque joueur. J'ai utilisé l'API BWAPI qui permet d'interfacer mon programme avec Starcraft. De plus, les séquences extraites ne représentent que les sept premières minutes de jeu alors qu'une partie complète dure généralement de 10 à 30 minutes. L'extraction des séquences est particulièrement longue, car on doit simuler les parties une après l'autre, une partie de 7 minutes prend environ 2 minutes à simuler.

Comparaison des séquences

Un des défis de travailler avec des plans est de trouver une mesure de distance entre deux plans. Pour le projet, j'ai décidé d'essayer l'alignement de séquence comme mesure de distance. Pour faire de l'alignement global de séquence, on doit d'abord déterminer une fonction de coût d'insertion et de suppression d'élément. Pour les builds orders, la fonction de coût ne dépend que des ressources nécessaires pour effectuer l'action. De plus, puisqu'une ressource est plus rare que l'autre, on pondère le coût de chaque ressource selon leur rareté. De plus, on ne peut aligner deux actions que s'ils sont identiques, sinon on doit faire une insertion ou une suppression. Pour trouver l'alignement global, j'ai utilisé un algorithme de programmation dynamique (Needleman–Wunsch).

Clustering

Avec une mesure de distance, on peut faire du clustering sur les builds orders recueillis où certains des clusters trouvés représentera un plan fréquent qui pourra être ajouté dans une librairie de plan.

Complete Link

J'ai choisi d'utiliser la technique de clustering hiérarchique Complete Link pour trouver les clusters de plans. Cet algorithme nécessite de calculer la matrice de distance de toutes les séquences. À l'initialisation de Complete Link, chaque plan est un cluster, à chaque itération on regroupe deux clusters ensemble jusqu'à ce qu'il n'y ait qu'un seul cluster. Cette technique forme un dendrogramme et il ne reste plus qu'à spécifier le nombre de clusters pour savoir à quel niveau couper notre dendrogramme.

Coefficient de silhouette

Le coefficient de silhouette permet de mesurer la qualité d'appartenance d'une donnée à un cluster. Cette mesure génère des valeurs entre -1 et 1 et plus la valeur est haute, plus la donnée est bien placée. En faisant la somme des coefficients de silhouette, on retrouve une mesure de qualité du clustering, plus la somme est haute, plus les clusters sont bien séparés.

Calcul du coefficient de silhouette :

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

Où a(i) représente la distance moyenne entre l'élément i et les éléments de son cluster, et b(i) représente la plus petite distance moyenne entre l'élément i et les éléments d'un autre cluster.

À partir du dendrogramme de Complete Link, on peut générer un graphique qui montre l'évolution de la qualité du clustering selon le nombre de cluster utilisé. Voir annexe 1 pour un exemple. À partir de ce graphique, on peut sélectionner à la main le nombre de clusters en regardant les « coudes » du graphique, c'est-à-dire les endroits où la somme augmente plus rapidement.

Modèle

Maintenant que nous avons un ensemble de clusters, nous devons en extraire un modèle qui pourra être utilisé dans notre librairie de plan. Pour ce faire, j'ai effectué un alignement multiple sur toutes les données d'un cluster pour générer une séquence qui représente le plan général du cluster.

Alignement multiple

Pour effectuer l'alignement multiple, j'ai décidé d'utiliser une technique d'alignement progressive deux à deux. Il est important de noter qu'il s'agit d'une technique approximative.

La méthode utilisée doit tout d'abord faire un alignement simple de toutes les paires de séquence pour former une matrice de distance. Par la suite on doit former un « guide tree » en rassemblant progressivement les groupes de séquence les plus similaires. Le « guide tree » est en fait un dendrogramme. Par la suite, en suivant l'ordre donné par le « guide tree », on effectue un alignement sur les deux séquences les plus similaires pour n'en former qu'une seule et l'on répète jusqu'à ce qu'il ne reste plus qu'une seule séquence. Cette séquence représente le plan général du cluster, par contre la séquence comportera des actions rarement utilisées, il faut donc élaguer les actions qui ne sont pas assez fréquentes parmi les séquences pour obtenir le modèle final.

Voici un exemple simple où l'on voit l'alignement multiple d'un cluster. La première ligne représente le modèle du cluster et les autres, les séquences contenues dans le cluster. Les colonnes en rouge représentent les actions qui ne sont pas assez fréquentes pour être conservé dans la séquence finale. Dans cet exemple, chaque valeur représente une action.

106, 007, 007, 007, 007, 007, 109, 007, 007, 111, *007*, 007, 007, 007, 007, 106, *007*, 109, 007, 007, *000*, 111, *007*, *109*, 000
106, 007, 007, 007, 007, 007, 109, 007, 007, 111, ___, 007, 007, 007, 007, 106, ___, 109, 007, 007, ___, 111, ___, ___, 000
106, 007, 007, 007, 007, 007, 109, 007, 007, 111, ___, 007, 007, 007, 007, 106, ___, 109, 007, 007, ___, 111, ___, ___, 000
106, 007, 007, 007, 007, 007, 109, 007, 007, 111, ___, 007, 007, 007, 007, 106, ___, 109, 007, 007, ___, 111, ___, ___, 000
106, 007, 007, 007, 007, 007, 109, 007, 007, 111, ___, 007, 007, 007, 007, 106, ___, 109, 007, 007, ___, 111, ___, ___, 000
106, 007, 007, 007, 007, 007, 109, 007, 007, 111, ___, 007, 007, 007, 007, 106, ___, 109, 007, 007, ___, 111, ___, ___, 000
106, 007, 007, 007, 007, 007, 109, 007, 007, 111, ___, 007, 007, 007, 007, 106, ___, 109, 007, 007, ___, 111, ___, ___, 000
106, 007, 007, 007, 007, 007, 109, 007, 007, 111, ___, 007, 007, 007, 007, 106, ___, 109, 007, 007, ___, 111, 007, ___, ___
106, 007, 007, 007, 007, 007, 109, 007, 007, 111, ___, 007, 007, 007, 007, 106, ___, 109, 007, 007, ___, 111, 007, ___, ___
106, 007, 007, 007, 007, 007, 109, 007, 007, 111, ___, 007, 007, 007, 007, 106, ___, 109, 007, 007, ___, 111, 007, ___, ___
106, 007, 007, 007, 007, 007, 109, 007, 007, 111, ___, 007, 007, 007, 007, 106, ___, 109, 007, 007, 000, 111, ___, ___, ___
106, 007, 007, 007, 007, 007, 109, 007, 007, 111, 007, 007, 007, 007, 007, 106, 007, 109, ___, 007, ___, 111, ___, ___, ___
106, 007, 007, 007, 007, 007, 109, 007, 007, 111, ___, 007, 007, 007, 007, 106, ___, ___, 007, 007, ___, 111, 007, 109, ___

Un des avantages à utiliser cette méthode pour faire de l'alignement multiple est que la matrice de distance et le « guide tree » sont déjà calculés pour faire le clustering hiérarchique, on économise donc beaucoup de calcul.

Complexité

Un désavantage de la procédure que j'ai utilisée est sa grande complexité. Voici donc la complexité de chaque partie de l'algorithme.

La variable *n* représente le nombre de séquence total, *c* le nombre de séquence dans un cluster et *l* le nombre d'actions maximales dans les séquences.

Alignement simple : $O(l_i * l_j)$

Où l_i et l_j représentent le nombre d'actions dans la première et la deuxième séquence respectivement.

Alignement multiple : $O(c * l^2)$

Car on doit faire *c* alignement simple.

Calcul de la matrice de distance : $O(n^2 * l^2)$

Car on doit faire un alignement simple pour chaque paire de séquence.

Génération du dendrogramme : $O(n^2)$

Représente le parcours de la matrice de distance

Somme des Coefficients de silhouette : $O(n^2)$

Pour calculer la somme des coefficients de silhouette pour un nombre de clusters fixe.

Algorithme complet : $O(n^3 + n^2 * l^2)$

La complexité de l'algorithme est surtout affectée par *n* calculs de somme de coefficient de silhouette et le calcul de la matrice de distance.

En plus de ces complexités, la sélection du nombre de clusters doit être faite manuellement pour l'instant. Aussi, la génération des séquences d'actions à partir de match enregistrée est un processus qui prend beaucoup de temps, soit 2 minutes par match.

Prochaine étape

Bien que le projet ait donné de bons résultats, il reste encore beaucoup de travail à faire.

Séquence temporelle

Pour ce projet, je n'ai pas pris en compte à quel temps les actions ont été effectuées, alors que le temps est un élément important de Starcraft. On devrait trouver une façon de prendre en considération cette information.
Un de problème est que la représentation d'un plan en séquence n'est pas parfaite, en effet un plan n'est pas toujours une séquence. Par exemple, dans un plan, certaines parties peuvent être exécutées en parallèle ce qui peut poser problème lorsque le plan est représenté par une séquence. Utiliser une approche qui tient compte du temps pourra donner de meilleurs résultats pour ce problème.

Plan partiel

Plutôt que de chercher un plan qui tient compte de toute la séquence, on pourra tenter de chercher des plans partiels, c'est-à-dire des séquences d'action fréquente de plus petite taille qui se trouve à l'intérieur des séquences initiales. Pour ce faire, on peut essayer le principe d'alignement local plutôt que l'alignement global.

Plan tactique

On pourra réutiliser le même raisonnement pour analyser des plans tactiques (microgestion). Par contre, les plans tactiques sont plus compliqués à représenter, car la position est très importante. Il est probable qu'un plan tactique ne pourra pas être traduit en séquence d'actions fixe.

Nouveau modèle

Il serait intéressant de représenter le plan d'un cluster par un modèle plus élaboré. Par exemple, plutôt que d'utiliser une séquence, on pourrait faire un arbre probabiliste où chaque nœud représente une action et chaque chemin représente une ou plusieurs des séquences du cluster.

Conclusion

En conclusion, j'ai réussi à produire des librairies de plan stratégique en utilisant des techniques d'alignement de séquence empruntées à la bio-informatique sur des séquences d'actions de match enregistré. Grâce à l'alignement de séquence, il est possible de générer une mesure de distance entre des plans ou des séquences d'actions, cette technique ouvre donc de nouvelles idées à essayer dans le domaine de la planification.

Annexe 1 : Sélection du nombre de cluster pour 57 Builds Orders de Terran