INF8215 : Intelligence artif.: méthodes et algorithmes

Rapport
Projet Quorridor

Équipe : Les_deux_canetons

Soumis par :
Jacob Brisson - 1954091
Guillaume Thibault - 1948612

Méthodologie

L'agent développé pour ce projet est basé sur la recherche adversairielle par l'utilisation de l'algorithme Minimax et d'heuristiques. L'algorithme Minimax a aussi été quelque peu modifié afin de pouvoir utiliser des heuristiques de pruning pour permettre une recherche plus approfondie et une méthode pour arrêter la recherche s'il ne reste plus beaucoup de temps à l'agent.

L'agent est séparé à 3 logiques différentes selon le stade de la partie. La première partie est les mouvements d'ouverture préenregistré. Lors de nos recherches [1], nous avons trouvé six séries de mouvement d'ouverture pour l'agent qui peut exécuter pour se placer dans une position stratégique. Ces mouvements ont alors été enregistrés et l'agent choisit de façon aléatoire parmi ces six ouvertures de jeu. Ces ouvertures varient entre 1 et 4 mouvements que le joueur essaie de réaliser et si celui-ci n'est pas en mesure, il utilise alors la logique suivante.

La deuxième partie logique de l'agent est l'algorithme Minimax quelque peu modifié que nous avons développé. L'algorithme Minimax incorporant la méthode de l'alpha bêta pruning a été modifié par l'ajout de minuteurs qui permet d'arrêter la recherche s'il ne reste plus grand temps à l'agent. Si c'est le cas, l'algorithme retourne alors que la meilleure action est de bouger sur le chemin le plus court. Nous avons aussi ajouté une condition contrôlée par une heuristique paramétrable pour évaluer si l'action vaut la peine ou non d'être explorée en profondeur. Cette action est alors évaluée par l'heuristique d'évaluation. À l'aide de cette fonctionnalité, l'agent évite de vérifier les murs très éloignés de l'autre agent qui ne lui bloquerait pas le chemin. La dernière modification apportée à l'algorithme Minimax est de permettre aux coups qui déplacent le joueur de pouvoir atteindre un niveau de profondeur afin de regarder si le coup peut se faire facilement bloquer. Notre algorithme Minimax utilise deux types d'heuristique comme mentionnés, un pour évaluer si le coup est bon ou non et un pour évaluer s'il est utile ou non d'explorer en profondeur l'arbre selon un coup et un pour faire l'évaluation des actions. Nous avons développé plusieurs heuristiques pour ces deux types qui sont utilisés à différents moments de la partie :

Évaluation :

- Heuristique aucun mur : Évalue la distance restante pour que l'agent atteigne son objectif. Plus celui-ci est loin, plus il pénalise.
- Heuristique fin de partie : Bloque l'autre joueur en maximisant le chemin minimal pour gagner.
- Heuristique général: Évalue le jeu selon plusieurs petites heuristiques (chemin du joueur, nombre de murs derrière lui, les chemins possibles et la différence des distances des joueurs pour gagner) en les combinant avec certains poids modelables.

- Pruning
    - Pruning de base : Évalue toutes les actions possibles.
    - Pruning proche joueur : Cette heuristique permet de considérer les mouvements et seulement les murs dans à une certaine distance du joueur.
    - Pruning bloquer adversaire : Cette heuristique permet de considérer les mouvements et les murs qui se trouvent sur le chemin de l'adversaire. Il est aussi possible de demander à l'heuristique de considérer notre chemin le plus court ainsi que les murs qui sont autour du joueur adverse.

Finalement, la dernière logique de l'agent est de retourner le chemin le plus court s'il ne reste plus de temps, si les deux joueurs n'ont plus de murs ou si une erreur s'est produite.

Résultats et Évolution de l'agent

Pour ce travail, nous avons fait plusieurs itérations de notre agent. À chaque nouvelle itération, plusieurs parties ont été réalisées avec nouvel agent contre une ancienne version afin de vérifier que ce nouvel agent est bien meilleur que le précédent. De plus, nous avons gardé plusieurs anciennes versions de notre agent afin de s'assurer que le dernier agent réalisé est bien meilleur de toutes nos anciennes versions.

La première version de l'agent est avec l'algorithme Minimax de base. Cet agent a beaucoup de possibilités de jeu alors l'arbre de recherche devient extrêmement grand rapidement. L'agent a alors besoin d'une profondeur maximum très petite soit de 0 ou 1. Lorsque cette profondeur est atteinte, une heuristique extrêmement simple basée sur la différence entre la distance de l'agent pour atteindre la victoire et celle de son adversaire est utilité pour estimer si l'action est bonne ou non. Pour améliorer cet agent, l'alpha bêta pruning est utilisée afin diminuer le nombre d'options possibles lors de la recherche en profondeur. Malheureusement, le nombre de possibilités d'action restant encore très élevé dès le premier niveau, l'algorithme peut seulement attendre une profondeur de 2, et il reste aveugle à bien de possibilité pouvant beaucoup le pénaliser. À ce stade, l'agent peut jouer une partie, mais n'est pas très intelligent et facilement battable.

La prochaine amélioration apportée est l'amélioration de l'heuristique utilisée lorsque la profondeur maximale est atteinte. Tout d'abord, nous avons amélioré l'ancienne heuristique pour prendre davantage en compte les chemins de notre agent. Il est intéressant de garder l'heuristique, car elle permet de donner une valeur intéressante à propos du chemin. Par la suite, nous avons créé une heuristique permettant de retourner une valeur basée sur le nombre de murs derrière. Finalement, nous avons ajouté une heuristique permettant de trouver le nombre de chemins possible que l'agente peut prendre, et ce en évaluant s'il vaut davantage la peine de retourner sur ses pas. Finalement, nous avons combiné tous ces heuristiques en leur donnant chacun un certain poids et nous avons aussi ajouté une clause permettant de pénaliser grandement l'action si celle-ci permet de faire gagner le joueur adverse dans les deux prochains tours sans que l'on puisse gagner avant. Cette heuristique

finale permet maintenant de battre l'ancienne heuristique à presque tous les coups et joue maintenant davantage comme un joueur réel.

La prochaine étape a été d'améliorer la recherche en profondeur de l'algorithme Minimax. Pour cela, nous avons légèrement modifié l'algorithme afin de pouvoir utiliser une heuristique de pruning pour enlever l'évaluation de certaines actions de façon intelligente et ainsi réduire la taille de l'arbre pour enfin permettre d'effectuer une recherche plus profonde. La première itération de cette heuristique de pruning fut de permettre à l'algorithme de considérer tous les mouvements ainsi que seulement les murs étant sur le chemin le plus court de l'adversaire lui bloquant le chemin. Par la suite, plusieurs options ont été ajoutées afin de pouvoir considérer de plus en plus d'action au fils que la partie avance, car moins d'actions sont possibles à ce moment et il est ainsi moins couteux d'explorer plus de possibilités pour une même profondeur. Les options ajoutées ont été de considérer les murs autour du joueur adversaire à une distance modulable et si ce mur bloque notre joueur. L'ajout de cette heuristique nous a permis d'exécuter Minimax de façon plus profonde tous au long de la partie, et ce en considérant de plus en plus de possibilités lorsque celle-ci approche de la fin. Cette heuristique a amélioré nos résultats de façon considérable et permet d'exécuter des jeux impressionnants, mais le retrait de l'évaluation de certaines actions rend l'agent aveugle à certains cas spéciaux qui peut le pénaliser grandement. Ce nouvel agent réussi alors à battre l'autre agent pour un peu plus de la moitié des parties.

L'étape suivante fut d'ajouter des jeux d'ouverture de partie pour les deux joueurs. En début de partie, trop de possibilités sont possibles et l'utilisation de façon aléatoire d'une ouverture permet de ne pas gaspiller de temps de recherche et de faire des coups qui amènent l'agent sur une bonne voie.

Pour améliorer davantage l'agent, nous avons conçu d'autres heuristiques pouvant être utilisées à divers moments dans la partie selon certaines conditions. La première heuristique fut une heuristique de permettant au joueur de considérer seulement les actions de mouvement et de mur autour des deux joueurs et la deuxième heuristique fut une heuristique pour l'évaluation à la profondeur maximale pour lorsque le jeu est considéré en fin de partie pour permettre de bloquer l'autre joueur en maximisant sont chemin minimal pour gagner.

Avec tous ces changements, l'agent obtient de bons résultats contre l'agent précédent, mais se fait battre par des versions beaucoup plus anciennes et aussi beaucoup plus simples de soi-même.

À ce moment, nous avons aussi testé l'agent sur le joueur aléatoire et le joueur greedy, nous avons vite réalisé que l'agent développé était en train d'overfit les agents précèdent. En effet, le nouvel agent performe alors très mal face à ces deux joueurs. Pour remédier à la situation, nous avons pris les fonctionnalités de l'agent développé et essayer plusieurs combinaisons d'heuristique pour différents stades de la partie, le tout en ajustant les valeurs paramétrables des heuristiques, en testant contre les trois agents, aléatoires, greedy et l'ancien agent. À l'aide de ces tests effectués contre ces agents qui jouent de façon extrêmement différente, nous

avons obtenu un agent qui permet de battre l'aléatoire et le greedy à tous les coups ainsi que l'ancien agent la majorité du temps.

Finalement, le dernier changement fut d'ajouter une contrainte pour que l'agent ne puisse pas manquer de temps, même si ce n'était pas le cas lors de nous test, pour être sûr de ne pas être disqualifié.

Pour ce qui est des résultats du tournoi Challonge, notre agent a réussi à gagner 3 de ses 5 tournois lors de la phase de poule. Malheureusement, celui-ci a seulement gagné 13 parties sur les 25 et ce ratio l'a placé en 4e place du groupe de poule ce qui ne lui a pas permis de passer au stage final. En consolation, notre agent peut se sentir fier d'avoir réussi à battre le meilleur joueur de notre poule qui s'est rendue en demi-final en gagnant 4 sur 5.

Discussion

L'avantage principal de notre conception par Minimax est que l'agent n'a pas besoin d'être entrainé. Ceci permet de rapidement changer l'agent sans devoir attendre et générer un grand nombre de parties avant qu'il puisse jouer avec de bonnes performances. L'agent et les heuristiques développées permet d'être très performant dans certaines situations, comme le début de partie pour bloquer l'adversaire et essayer de lui faire rebrousser chemin qu'au milieu de la partie en essayant toujours d'allonger le plus possible le chemin de l'adversaire. Un autre avantage principal de notre agent est celui qui est conçu pour utiliser le 5 minutes de façon à ce qu'il lui reste du temps pour pouvoir évaluer plusieurs coups en fin de partie. Cependant, si le temps alloué était changé, celui-ci perdrait son avantage, car la profondeur du Minimax a été choisie pour ne pas prendre trop de temps en début et milieu de partie.

Les autres désavantages de notre agent sont qu'il ne peut pas explorer toutes les possibilités et qu'il reste parfois aveugle à certains pièges de l'adversaire. Dans certains cas, notre agent peut avoir deux chemins possibles pour se rendre à destination et celui-ci peut en emprunter un sans bloquer l'autre, ce qui permet à l'adversaire de lui bloquer le passage juste avant de gagner. Il doit alors prendre le chemin arrière et il perd alors de précieux tours.

Référence

[1] Quoridor Strats, https://quoridorstrats.wordpress.com/category/strategy/openings/, consulté le 27 octobre.