Université Libre de Bruxelles

INFO-F-311

Intelligence Artificielle

Projet 2 : Recherche adversariale

Nieto Navarrete Matias , 502920

B-INFO

29 octobre 2023

1 Explication de BetterValueFunction

Dans `BetterValueFunction`, la méthode `transition` a été surchargée pour ajuster la manière dont la valeur d'un état est calculée après une transition d'état en fonction d'une action donnée.

— **Encourager la collecte de gemmes :** Pour encourager la collecte de gemmes, un bonus est accordé pour chaque gemme collectée. Ce bonus est calculé comme le produit du nombre de gemmes collectées et une valeur fixe de 1000. Cela incite fortement les agents à recueillir des gemmes puisque la valeur de l'état augmente de manière significative pour chaque gemme recueillie.

$$\texttt{gem\_bonus} = \sum \texttt{state.world\_state.gems\_collected} \times 1000$$

— **Pénaliser les mouvements excessifs :** Afin d'éviter des mouvements excessifs ou inutiles de l'agent, une pénalité est appliquée à chaque mouvement effectué. Cette pénalité est définie comme -1 pour toute action qui n'est pas "Stay". Cela signifie que si un agent se déplace, il reçoit une pénalité, ce qui encourage l'agent à réfléchir à deux fois avant de se déplacer sans raison valable.

$$\texttt{movement\_penalty} = \begin{cases} 0 & \text{if } \texttt{action} = \text{"Stay"} \\ -1 & \text{otherwise} \end{cases}$$

— La nouvelle valeur d'un état est alors la somme du bonus pour la collecte de gemmes et de la pénalité pour les mouvements. Cette valeur est assignée à l'état résultant après la transition.

$$\texttt{new\_value} = \texttt{gem\_bonus} + \texttt{movement\_penalty}$$

1.1 Résultats attendues

Avec cette nouvelle fonction d'évaluation, on s'attend à ce que les agents soient beaucoup plus motivés à recueillir des gemmes tout en évitant des déplacements inutiles. Par conséquent, pour une carte donnée, le nombre de nœuds étendus devrait être plus faible comparé à l'évaluation de base, car l'agent serait capable de trouver des solutions plus optimales en explorant moins d'états.

2 Comparaison du nombre d'états étendus

2.1 Observations pour le Monde 1

Nous observons que Minimax et sa version améliorée donnent les mêmes résultats. Ça laisse penser que la fonction améliorée ne change pas grand-chose ici.

Par contre, Alpha-Beta est plus efficace que Minimax, avec moins de nœuds étendus. Cette efficacité vient de l'élagage d'Alpha-Beta. Et avec une meilleure fonction d'évaluation, Alpha-Beta est encore plus performant. Notons que cette carte a seulement une gemme et aucun obstacle.

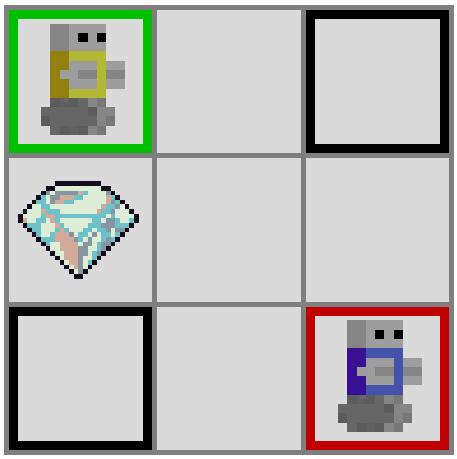
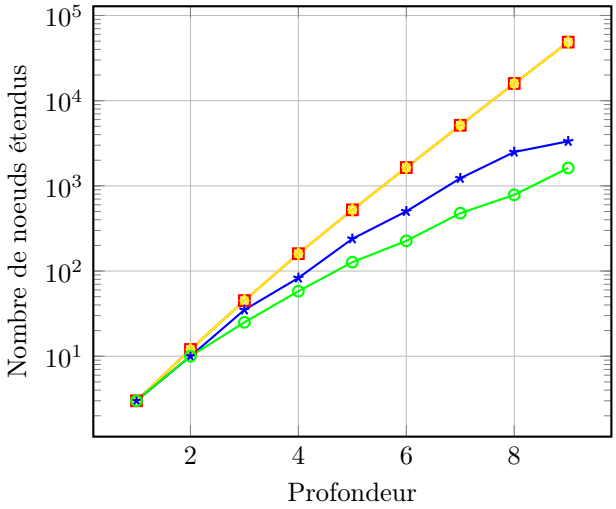


FIGURE 1 – Illustration du Monde 1

Comparaison des algorithmes Monde 1 (Voir Tableau 4.1)



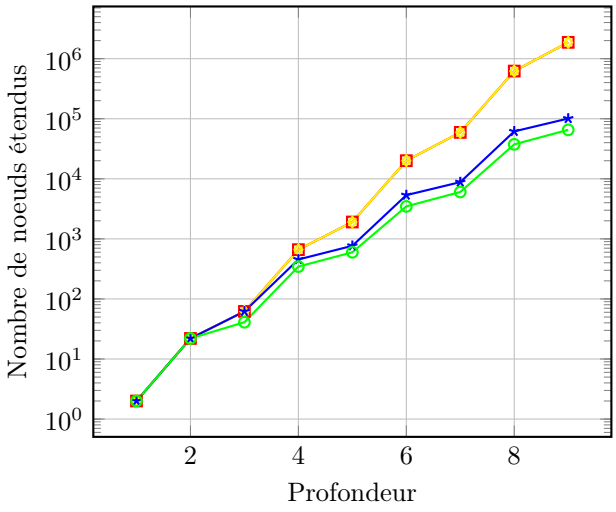
2.2 Observations pour le Monde 2

Comme dans le Monde 1, Minimax et sa version améliorée montrent le même résultat, sans différence notable.

En revanche, Alpha-Beta réduit le nombre de nœuds étendus. Sa version améliorée fait de même, mais cette réduction est moins marquée que dans le Monde 1. Cela peut être dû à la différence de la carte : ici, il y a plus d'obstacles et de gemmes.

FIGURE 2 – Illustration du Monde 2

Comparaison des algorithmes Monde 2 (Voir Tableau 4.2)



2.3 Observations pour le Monde 3

Dans le Monde 3, Minimax et sa version améliorée donnent les mêmes nombres. Donc, la version améliorée ne change rien ici.

Par contre, Alpha-Beta est meilleur que Minimax car il regarde moins de nœuds. Et quand on utilise la version améliorée d'Alpha-Beta, il y a très peut de différence ce qui peut être dû à la complexité de ce monde.

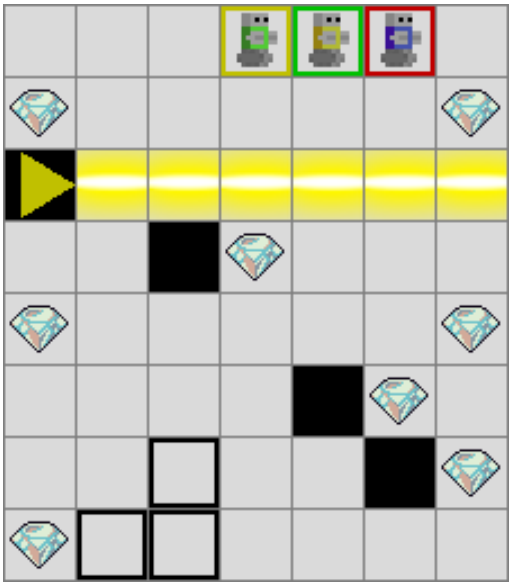
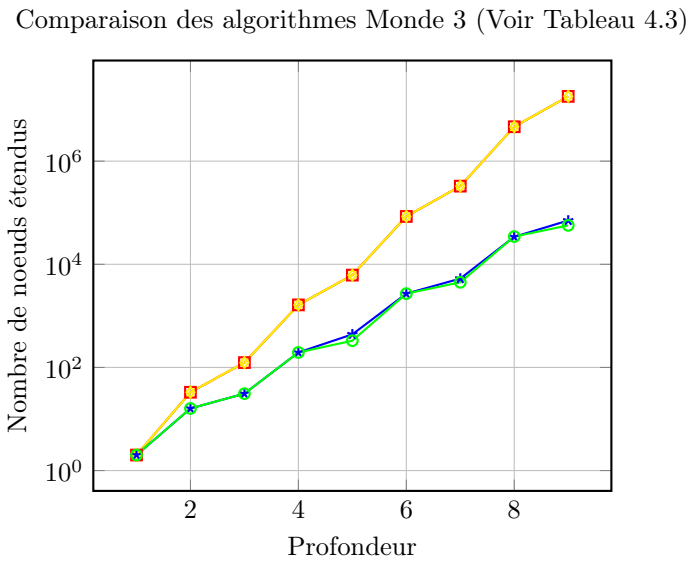


FIGURE 3 – Illustration du Monde 3



3 Conclusion

Notre `BetterValueFunction` a bien répondu aux attentes pour la méthode Alpha-Beta, en réduisant le nombre de nœuds parcourus. Cependant, lorsque le monde devient plus complexe, cette fonction d'amélioration réduit très peu, voire pas du tout, le nombre de nœuds étendus. Cela suggère que sa performance pourrait être affectée par la complexité du monde.

Néanmoins, la version optimisée de Minimax ne présente aucune réduction du nombre de nœuds. Cela s'explique par le fait que l'état terminal est indépendant du score. Ainsi, malgré une fonction d'évaluation améliorée, Minimax continue à explorer chaque branche de l'arbre de jeu.

ANNEXE

4 Tableaux des résultats

4.1 Tableau pour le Monde 1

Profondeur	Minimax	Minimax (BetterValueFunction)	Alpha-Beta	Alpha-Beta (BetterValueFunction)
1	3	3	3	3
2	12	12	10	10
3	45	45	35	25
4	160	160	83	58
5	523	523	239	127
6	1640	1640	502	226
7	5143	5143	1227	478
8	15920	15920	2496	784
9	48675	48675	3335	1622

TABLE 1 – Comparaison du nombre de noeuds étendus par profondeur pour différents algorithmes.

4.2 Tableau pour le Monde 2

Profondeur	Minimax	Minimax (BetterValueFunction)	Alpha-Beta	Alpha-Beta (BetterValueFunction)
1	2	2	2	2
2	22	22	22	22
3	62	62	62	41
4	662	662	454	346
5	1910	1910	767	598
6	20032	20032	5349	3475
7	59302	59302	8807	6043
8	622059	622059	61616	37499
9	1872134	1872134	100904	65389

TABLE 2 – Comparaison du nombre de noeuds étendus par profondeur pour différents algorithmes.

4.3 Tableau pour le Monde 3

Profondeur	Minimax	Minimax (BetterValueFunction)	Alpha-Beta	Alpha-Beta (BetterValueFunction)
1	2	2	2	2
2	33	33	16	16
3	124	124	31	31
4	1631	1631	195	195
5	6162	6162	438	330
6	84686	84686	2699	2699
7	327701	327701	5276	4474
8	4646185	4646185	34084	34493
9	18026586	18026586	70585	56819

TABLE 3 – Comparaison du nombre de noeuds étendus par profondeur pour différents algorithmes.