Etat de l’art des algorithmes d'intelligence artificielle utilisés dans les jeux de stratégie

[**Introduction 1**](#_heading=h.3znysh7)

[**Les algorithmes classiques pour les jeux de stratégie 2**](#_heading=h.2et92p0)

[Algorithme Minmax 2](#_heading=h.3dy6vkm)

[Algorithme Alpha Beta 2](#_heading=h.1t3h5sf)

[**Les Algorithmes de pathfinding 3**](#_heading=h.insxnujbgkyj)

[L’algorithme de Dijkstra 3](#_heading=h.1g0bzjc6kf91)

[L’algorithme A\* 3](#_heading=h.qtkjucoxxw89)

[Parcours en profondeur avec élagage 3](#_heading=h.1va4y3mi8oms)

[**Les algorithmes de deep learning 4**](#_heading=h.5rg50gvfajw)

[Apprentissage par renforcement 4](#_heading=h.wuzcegqyivbz)

[Réseaux de neurone convolutifs (CNN) 4](#_heading=h.5vrsltaz3wx4)

[**Conclusion 5**](#_heading=h.2s8eyo1)

[**Références 5**](#_heading=h.17dp8vu)

# 

# Introduction

"Dans les jeux, la meilleure décision n’est pas toujours celle qui vient immédiatement, mais celle qui anticipe plusieurs coups d’avance". C’est avec ce raisonnement que de nombreux scientifiques tels que John Von Neumann se sont mis à étudier la théorie des jeux [1]. En effet, après l'apparition de nombreux jeux de stratégie, qu’ils soient contre d'autres personnes ou non, les hommes ont alors cherché l'existence de stratégies leur permettant de maximiser leur chance de victoire. Certains jeux offrent la possibilité de toujours gagner, d’autres de ne jamais perdre ou juste d’augmenter nos chances de victoire. Cela dépend de la structure du jeu, des issues du jeu (gagner, perdre, match nul etc). Le jeu peut également faire s’affronter plusieurs joueurs ou bien faire affronter un robot adverse. Et contrairement aux jeux à information parfaite comme les échecs, l’aléatoire peut également entrer en jeu comme dans le blackjack.

C’est donc dans cette situation que les algorithmes de recherche de stratégies gagnantes voient le jour. Dans le contexte des jeux à deux joueurs sans aléatoire et à trois issues (victoire, défaite, match nul), on se place à un état *s* du jeu, alors on définit une stratégie gagnante comme une stratégie telle que, quels que soient les coups joués par son adversaire dans la suite, la victoire du joueur est garantie s’il suit cette stratégie. Dans notre cas, le jeu étudié étant Crossy Road, on ne peut trouver de réelle stratégie gagnante car ce jeu utilise l’aléatoire, le but sera donc juste de maximiser le score via l’utilisation d’algortihmes poussés.

# 

# Les algorithmes classiques pour les jeux de stratégie

Dans les jeux admettant des stratégies gagnantes (notamment les jeux à information parfaite), plusieurs algorithmes peuvent être mis en place. Dans le cas de Crossy Road ces mêmes algorithmes ne peuvent être implémentés tels quels et nécessitent des modifications, de plus, ils ne permettront pas d’être invincibles mais juste d’atteindre des scores très élevés.

### Algorithme Minmax

L’algorithme Minmax repose sur une évaluation des prochains coups possibles via un arbre de décision. Il est souvent utilisé dans un contexte de jeu à deux joueurs, à somme nulle et à information parfaite. L’algorithme va explorer de manière récursive les différents coups possibles des joueurs et va, via une fonction d’évaluation, essayer de maximiser ou de minimiser la sortie de cette fonction après un coup joué. Ici, si l'on considère que gagner vaut 1 et perdre vaut -1, l’algorithme va parcourir les coups possibles en supposant que le joueur adverse fasse les meilleurs choix (ceux l’approchant de la valeur 1), il devra alors minimiser les pertes. Pour crossy road il n’y a pas vraiment de joueur adverse mais une carte avec des informations, l’algorithme ne sera donc pas directement applicable sans modifications.

### Algorithme Alpha Beta

L’algorithme Alpha Beta est une optimisation de l’algorithme Minmax, en effet elle repose sur le même principe de maximisation et minimisation des valeurs des coups mais y ajoute aussi un processus d’élagage permettant de ne pas parcourir l’entièreté des coups si l’on sait que la branche actuellement parcourue ne mènera pas à une meilleure issue que la meilleure déjà trouvée. Pour cela, cet algorithme va utiliser deux valeurs limites: Alpha (valeur maximale atteinte par le joueur maximisant) et Beta (valeur minimale atteinte par le joueur minimisant). Cela va considérablement réduire le nombre de branches à parcourir, améliorant ainsi l’efficacité de la recherche [3]. Cet algorithme reposant essentiellement sur le même principe que Minmax, il ne sera donc pas de grande utilité mais l’idée de ne pas parcourir toutes les branches pourra être utile pour l’algorithme final

# 

# Les Algorithmes de pathfinding

L’utilisation d’algorithmes de pathfinding est essentielle lorsqu’il faut trouver un chemin optimal reliant un point à un autre en évitant des obstacles. Dans le cas de crossyroad le but du jeu est de trouver le chemin le plus long possible en avançant tout en évitant des obstacles. Ces algorithmes sont alors tout à fait approriés dans notre cas.

### L’algorithme de Dijkstra

L’algorithme de Dijkstra est un algorithme de recherche de plus court chemin dans un graphe orienté, ici les arêtes du graphe auront toutes la même pondération et notre graphe sera la grille. Il explore systématiquement toutes les possibilités et garantit de trouver le chemin le plus court, mais l’exploration exhaustive de tous les chemins d’un graphe avec |A| arêtes et |S| sommets a une complexité en O(|S|+|A|)log(|S|) avec l’implémentation en tas binaire et en O(|A|+|S|log(|S|) avec l’utilisation des tas de fibonacci [4]. Pour Crossy Road, cet algorithme pourrait être utilisé si l'on souhaite un déplacement optimisé minimisant les risques d’être bloqué. Cependant, sa nature exhaustive le rendrait moins efficace que son homologue A\* dans un jeu en temps réel.

### L’algorithme A\*

L’algorithme A\* repose sur le même principe que Dijkstra, c’est un parcours de proche en proche des sommets du graphe mais qui priorise l’exploration des chemins qui sont les plus suceptibles d’arriver à la destination. Cela est possible grâce à l'utilisation d’une fonction d’évaluation appelée l’heuristique. L’algorithme s’arrête dès lors qu’un chemin a été trouvé et celui-ci est le plus court. La recherche est donc non exhaustive, on a donc un algorithme plus efficace mais donnant moins de chemins valides. On rappelle qu’avec l’implémentation au programme des files de priorité (en tas binaire), la complexité de cet algorithme est en O(|A|log(|S|)). En utilisant une autre implémentation de files de priorité (les tas de Fibonacci), on peut faire descendre la complexité à O(|S|log(|S|) + |A|). [5]

### Parcours en profondeur avec élagage

Cet algorithme va calculer tous les chemins d’une profondeur fixée au préalable possibles. On utilisera alors un parcours en profondeur pour parcourir l’arbre des chemins possibles. Il est nécessaire d’avoir une profondeur fixée assez grande (>10 coups) car certains chemins de petite profondeur sont bons mais amène à une case qui est “bloquante”, en effet parfois aucun coup n’est possible mais la case étant accessible, le chemin est compté comme étant valide. Cependant, lorsque l’on augmente la profondeur, le nombre de calculs augmente considérablement plus. Un élagage des chemins permet alors de réduire le nombre de chemins à calculer. Pour cela, si deux chemins finissent de la même manière mais commencent différemment, si le premier chemin garde le poussin en vie, on ne va pas vérifier que le deuxième est un chemin est sans danger car on a déjà un chemin “valide” qui le fait. On va donc juste garder en mémoire les cases par lesquelles un chemin sans danger déjà calculé est passé et ignorer les chemins qui passent pas ces cases d’une autre manière. Le nombre de chemins à calculer est alors considérablement réduit et on peut donc augmenter un maximum la profondeur des chemins recherchés.

# 

# Les algorithmes de deep learning

Les techniques de deep learning ont beaucoup évolué ces dernières décennies et ont prouvé leur efficacité. Ces algorithmes permettent notamment de résoudre des problèmes complexes dans les jeux de stratégies.

### Apprentissage par renforcement

L’apprentissage par renforcement repose sur un système de récompenses données à des agent intelligents en fonction des actions qui entreprennent, leur but étant de maximiser ces récompenses. C’est alors en jouant de nombreuses parties que l’agent se perfectionne ce qui nécessite un long processus d'entraînement et de grandes capacités de stockage.

### Réseaux de neurone convolutifs (CNN)

Les réseaux de neurones convolutifs (CNN) sont utilisés afin d’extraire des caractéristiques visuelles, les interpréter puis prendre une décision [6]. Ceux-çi se prêtent bien à notre cas car les informations auxquelles il a accès sont exactement celles affichées à l’écran. Cependant il n’est pas nécessaire d’utiliser une telle IA car ces données peuvent directement être extraites et transmises via notre code.

# Conclusion

Les algorithmes classiques comme Minmax et Alpha-Beta sont adaptés aux jeux à information parfaite en 1vs1 mais beaucoup moins pour Crossy Road. Cependant les algorithme de pathfinding tels que Dijkstra et A\* sont plus pertinents lorsque le joueur doit trouver un chemin dans un environnement dynamique où les obstacles sont fatals. Aussi le développement de deep learning et de l’apprentissage par renforcement permettent à un agent d’apprendre à jouer de manière optimale mais cela nécessite d’importantes ressources. Ainsi, dans notre cas, le choix de l’algorithme penche plutôt vers l’utilisation de méthodes de pathfinding en utilisant un élagage afin de réduire nettement les temps de calculs.

# Références

[[1]](https://epubs.siam.org/doi/10.1137/S0097539791195543) « Theory of Games and Economic Behavior » John Von Neumann

[[2]](https://youtu.be/dQw4w9WgXcQ) « Conceptions et stratégies algorithmiques », A. Aho, J. Hopcroft, J. Ullman, Structures de données et algorithmes, Paris, InterEditions, 1987

[[3]](https://docs.lib.purdue.edu/cgi/viewcontent.cgi?article=1456&context=cstech) A. Aho, J. Hopcroft, J. Ullman (trad. de l'anglais), Structures de données et algorithmes, Paris, InterEditions, 1987, 450 p. (ISBN 2-7296-0194-5), « Conceptions et stratégies algorithmiques »

[[4]](https://fr.wikipedia.org/wiki/Formule_de_Luhn) « A short introduction to the art of programming » de Edsger W. Dijkstra, 1971,

[[5]](https://aoa0.github.io/pubs/icse22.pdf) ALGORITHMIQUE de DUNOD (Cormen leiserson rivest stein) 3e édition

[[6]](https://dl.acm.org/doi/pdf/10.1145/3658644.3691380) "7.6. Convolutional Neural Networks (LeNet)". Zhang, Aston; Lipton, Zachary; Li, Mu; Smola, Alexander J. (2024).