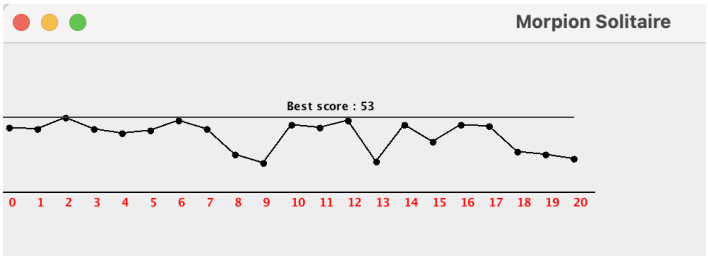
Rapport d'expérimentation

Nous avons essayé de réfléchir à un algorithme de recherche de la meilleure solution. Cependant nous n'avons pas réussi à coder tout l'algorithme auquel nous avons pensé. Cet algorithme est compliqué et nous avons réfléchi à faire de la récursivité pour le produire. Cet algorithme serait le suivant : pour chaque coup possible regarder combien de coups seront possible en n+1 (c'est-à-dire après avoir joué le coup possible combien de coups seront à nouveau possibles). Deux possibilités s'offrent maintenant à nous soit il existe parmi tous les coups possibles un unique coup qui produit le maximum de coup en n+1, dans ce cas on sélection ce coup, soit il existe parmi tous les coups possibles plusieurs coups qui atteignent le maximum en n+1 dans ce cas il faut les sélectionner et à partir de ses points il faut regarder le max en n+2. C'est là qu'intervient la récursivité car les étapes ensuite seront toujours les mêmes. Notre souci a été de nous dire comment faire pour ne pas impacter la grille lorsqu'on est dans le cas n+i. Pour combler ce souci nous avions pensé à faire une classe qui aurait pour but de copier l'état actuel de la grille courante et de pouvoir faire tourner l'algorithme dessus. Cependant, avec le temps imparti nous n'avons pas eu le temps de mettre en place cet algorithme. Nous avons mis en place plusieurs algorithmes plus ou moins basés sur notre idée avec l'intervention de l'aléatoire.

Nous avons mis en place plusieurs algorithmes :

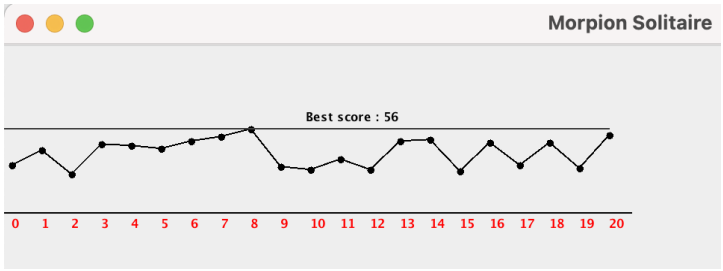
UpdateIA

On a commencé par créer une fonction naïve qui prend un point aléatoire parmi une liste de points possibles (coups possibles). Cette liste de points possible est mélangée puis le logiciel prend le premier point de cette liste.



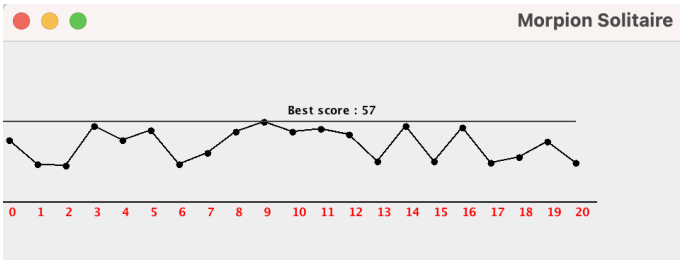
UpdateIA1

Cette méthode permet de sélectionner un point dans la liste des coups possibles (potentialMove) en fonction de son maximum dans une vision n+1. C'est-à-dire que pour tous les points (coups) possibles nous allons regarder combien de coups seront possibles à jouer après avoir joué ce point (coup). Cette méthode prend le premier point pour lequel elle rencontre le max (if(x>max)). Elle prend le premier car dès lors où elle a trouvé le maximum (sans qu'elle ne le sache encore) elle ne rentrera plus dans la condition if(x>max) qui permet de récupérer l'indice du point dans l'ArrayList potentialMove même si plusieurs points permettent d'atteindre le maximum en n+1.



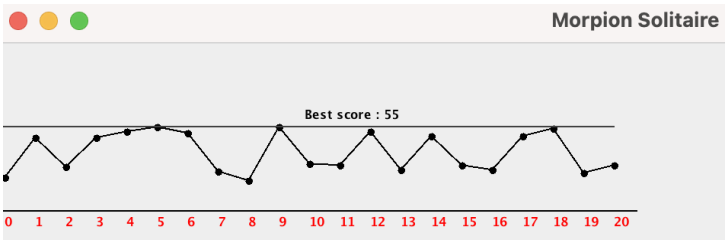
UpdateIA2

Cette méthode permet d'aller jusqu'à l'étape n+2. Cette méthode peut être appelée à n'importe quel moment du jeu lorsqu'il reste des coups possibles (points possibles dans potentialMove). Elle va tout d'abord récupérer le maximum de coups possible en n+1 pour chaque coup possible. Elle va ensuite remplir une ArrayList tampon des tous les coups possibles qui permettent d'atteindre le maximum en n+1. Ensuite si ce tableau tampon à une taille de 1 on sélection le point qui permet de jouer ce coup sinon on va regarder pour chacun de ses points leur maximum en n+2. La fonction remplit à nouveau une ArrayList dans laquelle forcément elle va faire un choix. Si la taille de cette nouvelle ArrayList est de 1 on renvoie le point sinon on mélange aléatoirement cette ArrayList et on prend le premier point de cette liste. L'algorithme s'arrête ici et on a donc une vision jusqu'à n+2 qui n'est souvent pas suffisante c'est pour cela que nous faisons un choix aléatoire mais pour réaliser cet algorithme entièrement il faudrait continuer à naviguer plus profondément jusqu'au moment où dans un le tableau tampon il nous reste qu'un élément.



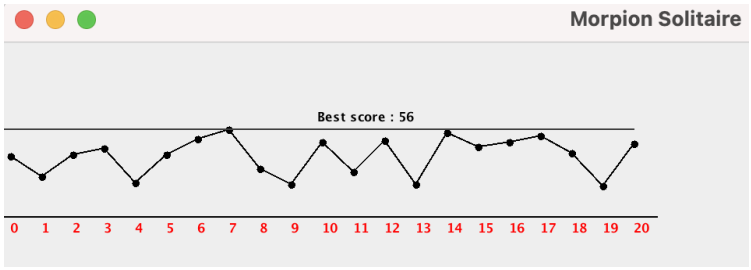
UpdateIA3 (disponible sur 5D uniquement)

Cette méthode parcourt la liste de coups potentiels et pour chacun elle calcule un score. Ce score est égal au nombre de point futures + 0.8 * le score lié la simulation de continuer la partie avec ce point.



UpdateIA4 (disponible sur 5D uniquement)

Cette méthode parcourt la liste de coups potentiels et pour chacun elle simule une partie. Néanmoins, elle stocke le plus gros score dans la classe Grid5D. Ici, on souhaite que le score max simulé soit au minimum égale à chaque coup. Dans un souci de temps de réponse, on a mis un timeout afin d'avoir des temps de réponses acceptables.



Comparaison entre ces 5 algorithmes

On a effectué 20 lancements sur 5 méthodes utilisant de l'aléatoire, néanmoins, on peut constater leur distribution. UpdateIA2 est celle qui a les meilleurs résultats.

Possibilité non implémentée (manque de temps) :

- Demander à la machine de continuer sa recherche jusqu'à ce qu'elle offre la grille contenant le max théorique.
- Demander à la machine de continuer sa recherche en gardant son score maximal temporaire au minimum. (Partiel)
- Réaliser un algorithme génétique qui pose le nombre de points théoriques autour de la croix jusqu'à ce que la combinaison soit parfaite et respecte les règles.
- L'algorithme récursive auquel nous avions pensé (explication en introduction de ce document)

Temps de travail : 50h-65h