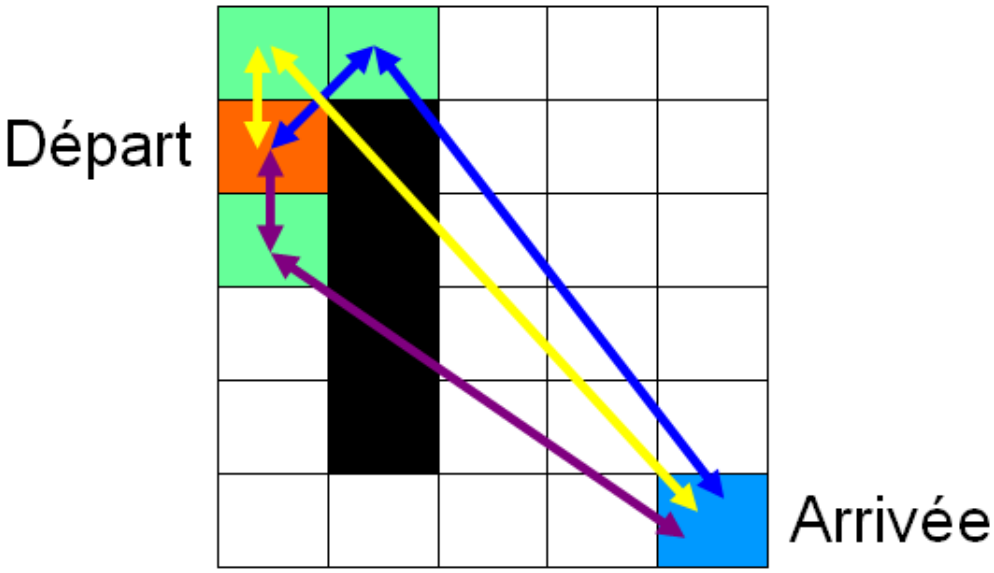
Documentation

Groupe 3 Bleu : Sokoban

Goron Nathan, De La Rosa Louis-David, Basset Emilien, Demé Quentin

Algorithme A*, fonctionnement



1 Principes de fonctionnement.

1.1 Présentation et objectif :

L'algorithme A* est un algorithme souvent utilisé pour trouver des chemins dans la résolution de jeux. Il est dans notre cas implémenté sur une grille de Sokoban. En dehors du sokoban où cela nous importe peu, il peut être interrompue pendant son utilisation est relancé plus tard. C'est un algorithme de type "anytime". Le but de A* est de trouver le chemin *le plus court* pour aller d'un point de départ, à un point d'arrivé. Pour cela l'algorithme garde une liste de toutes les étapes futurs que l'on appel "liste ouverte". Cette liste est établis par la fonction g(n). Ensuite, il choisit parmis cette liste une étape futur qui soit "à priori" la meilleure pour nous mener à l'objectif en un temps qui soit le plus court possible. Pour que cela puisse se faire, nous avons besoin d'une heuristique qui puisse nous aider à déterminer quelle est "à priori" la meilleure option. Une fois que cette option est choisis, l'algorithme passe à la "liste fermé".

2 Explication détaillée avec exemples.

2.1 Fonctions g et h :

A chaque étape de l'algorithme, les fonctions g(n) et h(n) sont exécutées pour n étant un point d'arrivé. Cela nous donne les informations suivantes.

- g(n) = Le nombre d'étapes pour aller du point de départ à l'arrivée n.
- h(n) = L'heuristique qui estime le coup pour aller de n à l'objectif final.
- f(n) = g(n) + h(n) : Le nombre minimum d'étapes si on choisit le point n.

2.2 Etapes détaillées de l'algorithme :

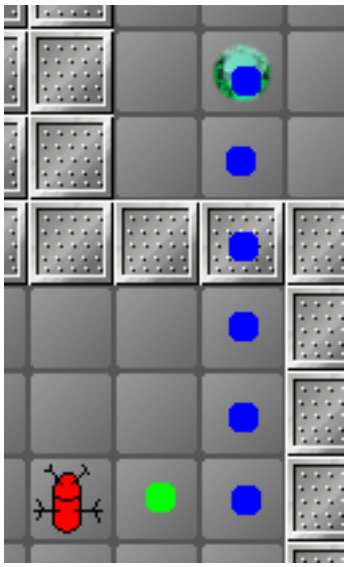
Voici les étapes de l'algorithme en langage algorithmique :
Soient : OPEN la liste ouvert
(La partie si dessous n'a pas pu être indentée correctement, attention donc !)
Ajouter DEPART à OPEN
Tant que OPEN n'est pas vide :
Choisir le N ayant le plus petit f(n)
Si N est l'objectif alors PASS
Déplacer N dans CLOSED
Pour chaque N' = PeuBouger(N, direction) :
g(N')=g(N)+1
h(N')
Si N' dans OPEN et nouveau N' n'est pas mieux, CONTINUE
Si N' dans CLOSED and nouveau N' n'est pas mieux, CONTINUE
supprimer tout N' étant dans OPEN et CLOSED (en même temps)
Ajouter N comme parent de N'
Ajouter N' à OPEN

FIN POUR
FIN TANT QUE
Si on arrive ici, alors il n'y a pas de solutions.

3 Le cas du Sokoban :

3.1 Cas général :

Dans le cas du sokoban, l'heuristique sera le minimum de déplacements pour atteindre l'objectif si un chemin direct est possible. Or puisque l'on se déplace uniquemment à la verticale ou à l'horizontale, cette donnée sera la somme des déplacements horizontaux et verticaux à réaliser.



Ce shéma présente le minimum de déplacements pour aller du point rouge au point d'arrivé.

Ainsi pour l'étape en vert :
g(n)= 1
h(n)= 6
f(n)= 1+6 = 7

3.2 Implémentation :

Tout d'abord, nous avons besoin d'une structure pour les noeuds :

class Node implements Comparable
public Node parent ;
public int move ;
public Location where ;
public int g ;
public int h ;
public int f()return g+h ;
public Node(Location where, Node parent, int move) ;
public int compareTo(Object o) ;

Tout d'abord, il est plus pratique de garder la liste OPEN triée pour que l'on puisse avoir facilement accées à notre meilleur option. Il faut donc s'assurer que lorsqu'on ajoute un terme à la liste, celle-ci soit triée automatiquement (Pourquoi ne pas utliser un ABR ? ?)
Ensuite il est efficace que nous puissons trouver des noeuds de la liste grâce à l'heure position sur la grille. Cela pour que nous puissons déterminer si 'n' est déjà dans OPEN ou dans CLOSE.

3.3 Note de l'auteur :

/* Les notes à suivre sont des notes de l'auteur qui n'utilise pas python */
On peut éventuellement mettre la liste des mouvements en cache(?) pour éviter de la recalculer à chaque fois.
L'algorithme peu être modifier pour calculer uniquement 'x' déplacement à un moment donné. Pour cela il faut garder OPEN et CLOSE entre les appels. Il faut se rappeler renvoyait à l'appel et calculer jusqu'à (g(n)t - g(n)t')>x,(soit un instant t) puis retourner la meilleure option de la liste OPEN.

4 Sources :

http ://www.talistreit.com/AStar.html

5 Annexe :

5.1 Shéma de fonctionnement :