Implémentation de A^* résolution du problème du taquin

Le but de ce TP est de résoudre le jeu du taquin en utilisant l'algorithme A*.

1 Le jeu du taquin

Le taquin se joue avec un petit dispositif comportant de petites pièces carrées mobiles formant un quadrillage. Une des pièces étant manquante, ces dernières peuvent glisser les une contre les autres. Le joueur peut ainsi modifier la configuration des pièces pour aller d'une situation donnée à une autre définie comme étant la solution.

Dans ce TP, nous utiliserons un taquin "3×3", dont les cases sont numérotées de 1 à 8, la case vide étant représentée par 0.

Pour résoudre ce problème nous utiliserons l'algorithme A* qui se prête bien au taquin. En effet, l'espace des situations du taquin forme un espace discrétisé représentable par un graphe. Et, d'autre part, le taquin se joue seul. L'algorithme A* va permettre de construire une lignée allant du noeud de départ au noeud solution.

Noeud de départ :

2	8	3
1	6	4
7	0	5

Noeud solution :

1	2	3
8	0	4
7	6	5

2 Rappels sur A*

L'algorithme A* suppose l'existence de 2 listes de noeuds, une appelée OUVERT l'autre FERME. la liste OUVERT comporte tout les noeuds en attente de traitement, et la liste FERME regroupe tout les noeuds traités. L'algorithme A* tente de minimiser une fonction f appelée fonction d'évaluation. Cette fonction est la combinaison linéaire de 2 termes, un terme évaluant la distance au but noté h, et un deuxième, noté g, indiquant la distance depuis le point départ. Cette fonction s'écrit alors $f = g + h$. Évidemment ces fonctions sont parfois difficiles à déterminer, il est possible d'utiliser une fonction estimée en particulier pour la fonction heuristique h. Cette estimée, notée h^*, doit respecter le critère suivant pour permettre la convergence : h^* ¡ h pour tout les noeuds. Choisir une valeur nulle permet de trouver toujours cette fonction, mais impose alors une recherche en largeur d'abord. La fonction d'évaluation peut aussi comporter un terme de pondération w, elle s'écrit alors : $f^* = (1 - w).h^* + w.g$. Ce terme permet de passer d'une recherche en profondeur d'abord ($w = 0$) à une recherche en largeur d'abord ($w = 1$).

3 Heuristique proposée

On se propose d'utiliser comme heuristique **le nombre de cases mal placées.**

4 L'algorithme

L'algorithme A*

Placer le noeud initial n dans OUVERT, $f^*(n) = 0$

tant que OUVERT n'est pas vide **faire**

1. Rechercher dans Ouvert le noeud ayant le plus faible F^*. Ce noeud, appelé m est placé dans FERME.
2. **pour** chaque fils, notés f_i, de m **faire**
   (a) mettre à jour le lien de parenté de f_i vers m
   (b) évaluer le noeud f_i : **si** f_i = but **alors** sortir
   (c) rechercher une occurrence de f_i dans OUVERT et FERME
   (d) **si** aucune occurrence de f_i est trouvée, ni dans OUVERT ni dans FERME **alors** le mettre dans OUVERT
   (e) **si** une occurrence, notée k, est trouvée dans OUVERT **et** $F^*(k) < F^*(f_i)$ **alors** ne pas rajouter f_i à OUVERT **sinon** supprimer k de OUVERT et rajouter f_i à OUVERT
   (f) **si** une occurrence, notée k, est trouvée dans FERME et $F^*(k) < F^*(f_i)$ **alors** ne pas rajouter f_i à OUVERT **sinon** supprimer k de FERME et rajouter f_i à OUVERT

Un des points les plus intéressant de l'algorithme est qu'un noeud déjà évalué (dans FERME) peut très bien être réévalué, il faudrait alors remettre à jour tous ces fils, ceci est fait en renvoyant ce noeud dans la liste des noeuds à traiter, ses fils étant alors éventuellement recréés et réévalués (voir point 2f de l'algorithme).

5 Implémentation

ATTENTION : pour rendre le code plus généralisable, on souhaite séparer la gestion A* de la génération des états. On propose le diagramme de collaboration suivant :