**Exercice 3**

**Méthodes de résolution heuristiques**

Les méthodes exactes sont souvent obsolètes face à des problèmes de grande taille, car très gourmandes en ressources. On peut alors utiliser certaines informations relatives au problème appelées **“informations heuristiques”** qui serviront à guider la recherche.

Certaines heuristiques réduisent considérablement l’effort de recherche mais ne garantissent pas de trouver le chemin de moindre coût, d’où l’importance de bien la choisir. En pratique on essaye de minimiser/maximiser une combinaison du **coût du chemin** et du **coût de la recherche** requise pour arriver au bout ce chemin.

A étoile (A\*) est un des algorithmes de recherche basé heuristique des plus connus et des plus utilisés. Le principe général de la recherche reste le même que celui de DFS et BFS. La différence réside dans la gestion de la liste des nœuds OUVERTE.

|  |
| --- |
| Algorithme A\*  **Entrée :**   * S : état initial; * P : tableau représentant l’instance I, * C(ni , nj ): le coût d’une action; * h : fonction heuristique ;   **Sortie:**   * un partitionnement de P en deux sous-ensembles à somme égale sinon échec ;   **Var :**   * Ouverte : liste de nœuds représentant les étapes d’instanciation triée selon la fonction f et initialement vide ; * Fermée : file de nœuds initialement vide ;   **Début**   1. insérer le nœud initial s dans la liste **Ouverte** avec f(s) := g(s) + h(s) ; 2. **si** (**Ouverte** = ∅) **alors** **échec** **sinon** continuer ; 3. retirer *n*, le premier nœud de la liste **Ouverte** et l’enfiler dans **Fermée** ; 4. si *n* n’a pas de successeur alors aller à 2 ; 5. **Pour chaque** successeur ni de n : 6. **Faire** f(ni ) := g(xi ) + h(ni); 7. insérer ni avec f(ni) dans **Ouverte** selon l’ordre croissant de f; 8. **Fait** créer un chainage de ni vers n; 9. **si** parmi les successeurs il existe un état final alors **succès** : la solution est le partitionnement à différence minimale ; 10. **sinon** aller à 2 ;   **Fin** |

**A faire :**

Il vous est demandé de :

* Proposer une heuristique h( ) et une fonction de coût g( ) pour le problème de partitionnement.
* Implémenter l’algorithme de recherche heuristique “A\*” pour la résolution du problème de partitionnement.
* Testez votre algorithme avec différentes tailles du problème.
* Illustrer vos résultats numériques par un graphe. Que remarquez-vous ?

Bon courage.