Cours 2 – Recherche Heuristique dans les graphes d'états

Master IAD – RP (M1) – Patrice Perny

LIP6 – Université Paris 6

Plan

1. Formulation du problème
2. Algorithme A*
3. Terminaison et correction
4. Exemples

Notations

On considère un graphe $G = (N, A)$ tel que :

- N est l'ensemble des noeuds (sommets) du graphe représentant les états possible du problème
- n_0 : noeud initial (état initial du problème)
- $\Gamma \subset N$: ensemble des noeuds buts (états que l'on cherche à atteindre)
- $S(n) \subset N$: noeuds successeurs du noeud n (ensemble des noeuds que l'on peut atteindre à partir du noeud n à l'aide d'une action admissible). On suppose que $|S(n)|$ est fini.
- $A = \{(n, p), n \in N, p \in S(n)\}$ ensemble des arcs du graphe (défini implicitement par la relation successeur)
- G est muni d'une valuation $k : A \to \mathbb{R}^+$, où $k(n, m)$ représente le coût de l'arc (n, m).

Formulation du problème

Problème : de recherche dans un graphe d'états

Déterminer, s'il en existe, une séquence d'états $n_0, n_1, \ldots, n_r = \gamma$ telle que :

I) $\gamma \in \Gamma$

II) $\forall i = 1, \ldots, r, (n_{i-1}, n_i) \in A$

III) $\sum_{i=1}^{r} k(n_{i-1}, n_i)$ est minimale sur l'ensemble des séquences vérifiant I) et II).

Il s'agit donc de rechercher un *chemin solution* (chemin allant de n_0 à Γ) de coût minimum dans un graphe G défini implicitement.

Une procédure générale de recherche

PROCEDURE Recherche(G, n_0, Γ, pere)

$O \leftarrow \{n_0\}$; $F \leftarrow \emptyset$; $trouve \leftarrow false$;
tant que $O \neq \emptyset$ et $trouve = false$ faire
    $n \leftarrow choix(O)$
    si $n \in \Gamma$ alors
        $trouve \leftarrow true$
        renvoyer la solution trouvée
    sinon
        développer(n) /* engendrer $S(n)$ */
        $O \leftarrow O \setminus \{n\}$; $F \leftarrow F \cup \{n\}$
        si $S(n) \neq \emptyset$ alors
            $\forall m \in S(n)$ décider si $pere(m) \leftarrow n$
            $O \leftarrow O \cup S(n)$
        fsi
    fsi
ftq

Contrôle de la recherche

fonctions : choix et décider

Recherche totalement ordonnée par une fonction $f : N \rightarrow \mathbb{R}$

Exemple 1 : *fonction profondeur* :

$f(n_0) = 0$
si $pere(n) = m$ alors $f(n) = f(m) + 1$

si $choix(X) = \arg\min_{x \in X} f(x)$ alors recherche en largeur d'abord

Exemple 2 : *fonction d'évaluation* :

$f(n)$ retourne une évaluation du coût potentiel des meilleurs chemins solution passant par n

Fonction d'évaluation

- $k^*(n, m)$ coût d'un chemin optimal de n à m ($+\infty$ si pas de chemin)
- $g^*(n) = k^*(n_0, n)$ coût du meilleur chemin arrivant en n
- $h^*(n) = \min_{\gamma \in \Gamma} k^*(n, \gamma)$ coût du meilleur chemin allant de n vers un noeud but
- $f^*(n) = g^*(n) + h^*(n)$ coût du meilleur chemin solution passant par n

  On va estimer $f^*(n)$ par $f(n)$ défini comme suit :

$$f(n) = g(n) + h(n)$$

- g(n) : coût du meilleur chemin déjà connu pour arriver en n (approximation par excès de $g(n)$)
- h(n) : heurisitique estimant, généralement de manière optimiste, la valeur de $h^*(n)$

Définitions relatives aux heuristiques

Soit $h : N \to \mathbb{R}$ une fonction heuristique. On dira que :

1. h est une heuristique *monotone* ssi :

$$\forall n \in N, \forall m \in S(n),\ \ h(n) - h(m) \leq k(n, m)$$

2. h est une heuristique *coïncidente* ssi :

$$\forall \gamma \in \Gamma,\ \ h(\gamma) = 0$$

3. h est une heuristique *minorante* ssi :

$$\forall n,\ \ h(n) \leq h^*(n)$$

4. h est une heuristique *mieux informée* que h' ssi :

$$\forall n \in N \setminus \Gamma,\ \ h'(n) < h(n) \leq h^*(n)$$

Obtention d'heuristiques minorantes (1/2)

Proposition

Toute heuristique monotone et coïncidente est minorante

Preuve

Soit $n, m_1, m_2, \ldots, m_q, \gamma$ un chemin optimal de n vers le but :

$$\begin{array}{ccc} h(n) - h(m_1) & \leq & k(n, m_1) \\ h(m_1) - h(m_2) & \leq & k(m_1, m_2) \\ \vdots & \leq & \vdots \\ h(m_q) - h(\gamma) & \leq & k(m_q, m_\gamma) \\ \hline h(n) - h(\gamma) & \leq & h^*(n) \end{array}$$

Comme h est coïncidente alors $h(\gamma) = 0$, donc $h(n) \leq h^*(n)$

$\square$

Obtention d'heuristiques minorantes (2/2)

Heuristiques obtenues par relaxation du problème

(on simplifie le pb en oubliant des contraintes pour calculer simplement une borne inf du coût pour atteindre le but)

Exemple 1 : distance "à vol d'oiseau" dans un problème de planification de trajectoire.

Exemple 2 : nb de dames à placer dans le problème des *n* dames.

Exemple 3 : distance rectilinéaire au but dans le problème du taquin.

ALGORITHME A*

PROCEDURE Recherche A*($G, n_0, \Gamma, f, g, h, pere$)

$O \leftarrow \{n_0\}$; $F \leftarrow \emptyset$; $g(n_0) \leftarrow 0$; $n \leftarrow n_0$
tant que $O \neq \emptyset$ et $n \notin \Gamma$ faire
  $O \leftarrow O \setminus \{n\}$; $F \leftarrow F \cup \{n\}$
    pour tous $m \in S(n)$ faire
      si $m \notin O \cup F$ ou $g(m) > g(n) + k(n, m)$ faire
        $g(m) \leftarrow g(n) + k(n, m)$
        $f(m) \leftarrow g(m) + h(m)$
        $pere(m) \leftarrow n$
        ranger m dans O par $f \uparrow$ et $g \downarrow$
      fsi
    finpour
  si $O \neq \emptyset$ alors $n \leftarrow first(O)$
ftq

A la fin si $O = \emptyset$ alors pas de solution sinon *pere* fournit le chemin solution.

Terminaison d'A* (le cas de G fini)

Proposition (P1)

Pour toute heuristique h à valeurs positive, et pour tout graphe G **fini** *admettant au moins un chemin fini de n_0 à Γ, A^* s'arrête au bout d'un temps fini.*

Idée de la preuve Nb fini de sommet. On ne peut ouvrir deux fois un sommet que si on trouve un meilleur chemin. Pas de circuit absorbant donc on ne peut emprunter les circuits qu'une seule fois. Au pire, pour un noeud n, tous les chemins menant à n sont engendrés. En fait dès que le meilleur arrivant en n est trouvé, il coupe tous les autres et le noeud n arrive en tête de O et est développé puis passe définitivement dans F. Développant ainsi les noeuds ouverts puis les fermant pour de bon au bout d'un temps fini, on progresse strictement vers le noeud but et on s'arrête.

□

Terminaison d'A* (le cas de G infini)

Proposition (P2)

Pour toute heuristique h à valeurs positives, et pour tout graphe G **fini ou infini** *admettant au moins un chemin fini de n_0 à Γ, tout chemin* optimal *de n_0 à Γ possède au moins un noeud n_i tel que : $g(n_i) = g^*(n_i)$.*

Preuve : Soit $(n_0, n_1, \ldots, n_q = \gamma)$ un chemin optimal de n_0 au but. Au départ n_0 est dans O, puis lors de son développement n_0 passe dans F et c'est n_1 qui entre dans O etc. A l'itération courante, soit n_i le premier noeud de la séquence à être ouvert, alors le sous-chemin $(n_0, n_1, \ldots, n_i)$ est connu. Comme c'est un sous-chemin d'un chemin optimal, il est optimal pour arriver en n_i (Principe de Bellman) et donc $g(n_i) = g^*(n_i)$.

□

TERMINAISON DE A* (LE CAS DE G INFINI)

PROPOSITION (P3)

Pour tout graphe G **fini ou infini** *admettant au moins un chemin fini solution $(n_0, n_1, \ldots, n_q = \gamma)$ et pour lequel tout chemin de coût borné supérieurement admet un nb fini d'arcs, pour toute heuristique h à valeurs positives telle que $h(u_i) \leq M$ pour $i = 0, \ldots, q$, alors A^* s'arrête au bout d'un temps fini en fournissant un chemin de n_0 à Γ.*

Preuve :

EXISTENCE D'UN CHEMIN DE COÛT MIN DANS G.

G admet un chemin solution de longueur finie. Soit k sont coût et soit $P(k)$ l'ensemble des chemins solutions de coût inférieur ou égal à k. Comme le coût de tout chemin $P(k)$ est borné, ces chemins admettent tous un nb fini d'arcs. De plus chaque noeud de ces chemins admet un nb fini de successeurs. Donc $P(k)$ est un ensemble fini. La fonction coût admet donc un minimum sur $P(k)$ qui est $f^*(n_0)$ le coût du chemin optimal.

Focalisation sur un sous-graphe fini G' de G.

Soit G' le sous-graphe de G restreint aux noeuds $G = \{n \in N, g^*(n) + h(n) \leq f^*(n_0) + M\}$. G' est fini (chacun de ses noeuds est atteint depuis n_0 par au moins un chemin fini). De plus, G' contient au moins un chemin solution de G. En effet, sur un tel chemin $(n_0, n_1, \ldots, n_q = \gamma)$ on a : $g^*(n_i) \leq g^*(\gamma) = f^*(n_0)$ et donc $g^*(n_i) + h(n_i) \leq f^*(n_0) + M$ donc chaque noeud du chemin est dans G'.

A* ne développe aucun noeud extérieur à G'. Supposons en effet qu'un noeud n qui n'est pas dans G' soit ouvert à l'itération courante. On aurait : $f(n) = g(n) + h(n) \geq g^*(n) + h(n) > f^*(n_0) + M$ par définition de G'. Or d'après P2, il existe à cette itération un noeud ouvert n_i sur un chemin optimal avec $g(n_i) = g^*(n_i) \leq f^*(n_0)$. On a alors $f(n_i) = g(n_i) + h(n_i) \leq f^*(n_0) + M < f(n)$. Donc n ne sera pas le meilleur élément de O et ne sera pas ouvert à cette itération.

Ceci montre que la recherche est restreinte au sous-graphe fini G' qui comporte au moins une solution optimale

$\square$

Admissibilité de A* (1/2)

Proposition (P4)

Si l'heuristique h est minorante alors à la fin de chacune des itérations d'A^*, le noeud n en tête de l'ensemble O est tel que : $f(n) \leq f^*(n_0)$.

Preuve : Par définition $f(n) = \min\{f(o), o \in O\}$. D'après P2 à tout moment il existe dans O un noeud n_i sur un chemin optimal tel que $g(n_i) = g^*(n_i)$. Comme $h(n_i) \leq h^*(n_i)$ on a : $f(n_i) \leq f^*(n_i) = f^*(n_0)$. Comme $f(n) \leq f(n_i)$ on a donc $f(n) \leq f^*(n_0)$.

□

Admissibilité de A* (2/2)

Proposition (P5)

Si l'heuristique h est minorante alors l'algorithme A^* est admissible, i.e. il s'arrête en fournissant un chemin optimal entre n_0 et Γ.

Preuve : On a vu que lorsqu'il existe un chemin solution, l'algorithme termine. Il finit par rencontrer un noeud but γ ouvert tel que : $f(\gamma) \leq f^*(n_0)$. Ce noeud est nécessairement l'extremité terminale d'un chemin de coût minimal.

$\square$

Remarques

Remarque 1 :
Si h est minorante alors le sous graphe G' auquel se limite la recherche est restreint aux seuls noeuds de l'ensemble :

$$\{n \in N, g^*(n) + h(n) \leq f^*(n_0)\}$$

Remarque 2 : Si h est monotone alors tout noeud développé par A* est tel que $g(n) = g^*(n)$. Il s'ensuit que le noeud n ne peut être développé qu'au plus une fois.

Remarque 3 : On peut montrer que la complexité d'A* est en $O(N^2)$. Ceci n'est toutefois pas toujours suffisant pour résoudre les problèmes pratiques efficacement dans la mesure où N peut être très grand (nb de noeuds).

(e.g. taquin : N = 362880 ; Rubik's cube : N = $4.3\ 10^{19}$)