Résolution de problèmes

Patrice Perny

LIP6 - UPMC

Informations relatives à l'UE RP

Equipe pédagogique

- Cours : Patrice Perny, Evripidis Bampis
- TD/TME : Nawal Benabbou, Thibault Lust

Horaires

- Cours : vendredi 13h15-15h45 salle 54-55, 203
- TD/TME Groupe 1 : jeudi 8h30-12h45, salle 23-24, 201,
- TD/TME Groupe 2 : vendredi 8h30-12h45 salle 13-14 109,

Evaluation

- Examen réparti 1 : 42.5 %
- Examen réparti 2 : 42.5 %
- Projet/TME : 15 %

Page du cours : http ://www-poleia.lip6.fr/∼perny/ANDROIDE1516/RP

Plan du cours MLA

Partie I (P. Perny)

- Introduction à la résolution de problèmes en IA
- Recherche heuristique dans les graphes d'états
- Satisfaction de contraintes I
- Satisfaction de contraintes II
- Introduction à la recherche locale

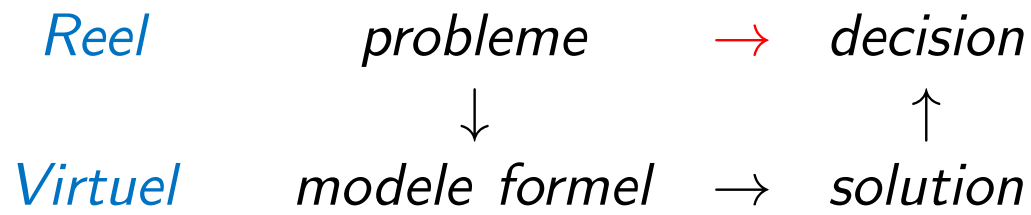
Partie II (E. Bampis)

- Recherche heuristique en RO,
- algorithmes approchés ...

INTRODUCTION

Modélisation

Démarche de modélisation :



Outils pour la modélisation :

- variables, contraintes
- graphes

Outils pour la résolution :

- algorithmes exacts
- algorithmes approchés avec garantie
- heuristiques

Principale difficulté (1)

Nature combinatoire des solutions potentielles

solution potentielle : combinaison de décisions élémentaires

$$sol = d_1 \circ d_2 \circ \ldots \circ d_n$$

exemple : $x = (x_1, \ldots, x_n)$

même si $x_i \in \{0, 1\}$ on a 2^n solutions potentielles
Le nb de solutions croît exponentiellement avec la taille du pb

exemple 1 : pb de la table à 8 places

exemple 2 : voyageur de commerce
(n ! solutions, 20 villes, ... 300 000 siècles)

Importance de la modélisation

Exemple 3 : pb de l'échiquier et des dominos

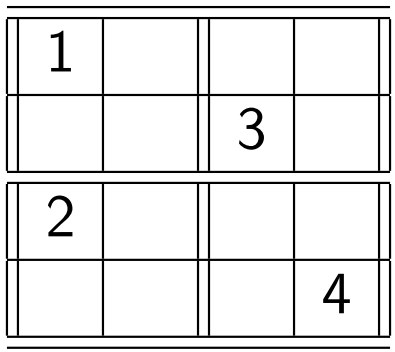
Exemple 4 : pb des 4 cavaliers

Difficulté de la modélisation :

- compacité de la représentation
- aptitude à représenter les contraintes
- sensibilité par rapport à l'ajout de nouvelles contraintes

Modélisation du Sudoku

Exemple 5 : pb du mini-sudoku



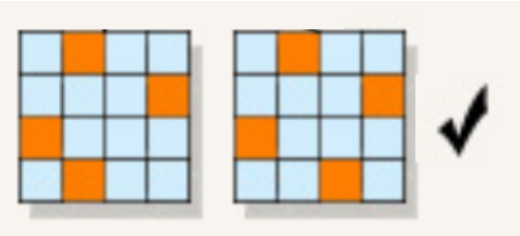
$4^{16} = 4\ 296\ 967\ 296$ solutions potentielles (pas toutes admissibles)

si on intègre les contraintes en ligne $24^4 = 331\ 776$ solutions (pas toutes admissibles)

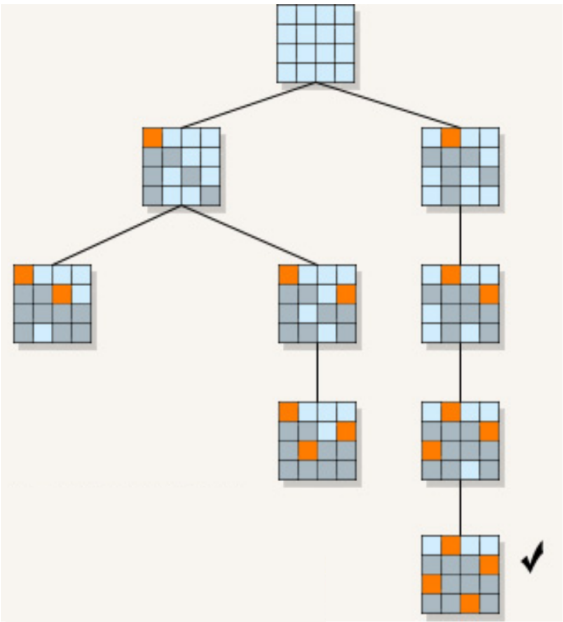
Modélisation

- n^3 variables binaires x_{ijk}
- n^2 variables $x_{ij} \in \{1, \ldots, n\}$
- écriture des contraintes

Problème des n dames

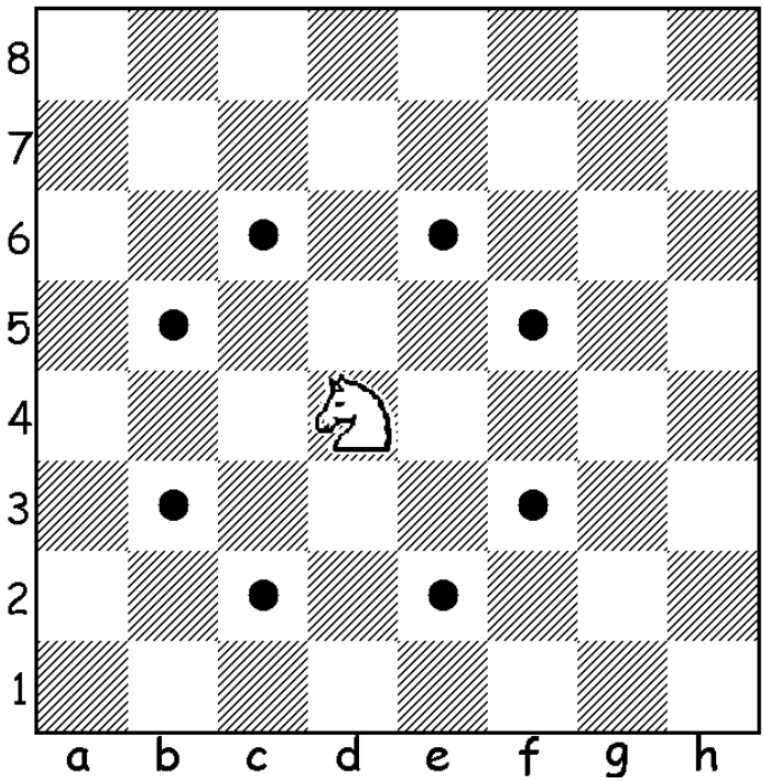


- x_{ij} variables binaires (existence d'une dame)
- x_i variables de domaine $\{0, \ldots, n\}$
- permutations de $\{1, \ldots, n\}$
- graphe d'état

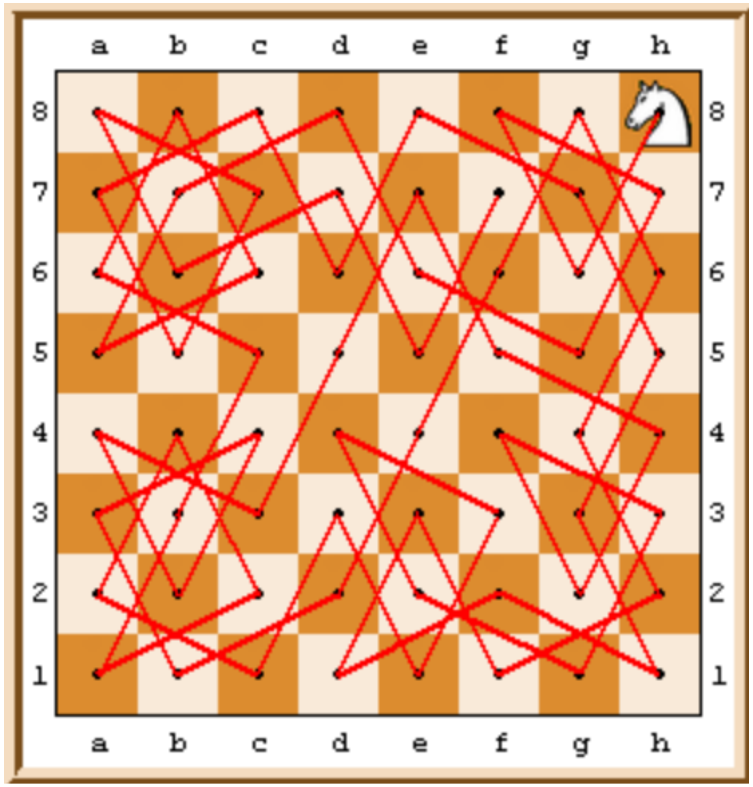


Le problème du cavalier hamiltonien

On pose un cavalier sur une case d'un échiquier. Le problème du cavalier hamiltonien consiste alors à effectuer des déplacements de cavalier successifs de façon à parcourir une fois et une seule chaque case de l'échiquier sans jamais repasser sur la même case.



Un exemple de solution



Modélisations

1. chemin ou circuit hamiltonien dans un graphe
2. problème de satisfaction de contraintes sur domaines finis

Soient (x_i, y_i) les abscisses et ordonnées de la ième position du cavalier sur l'échiquier dans le parcours recherché, i=0, . . . , 64.

Domaines des variables : $x_i \in \{1, \ldots, 8\}$, $y_i \in \{1, \ldots, 8\}$

Contraintes :

Si le cavalier doit revenir à la position initiale : $x_0 = x_{64}, y_0 = y_{64}$

Déplacements légaux du cavalier : $\forall i = 1, \ldots, 64$,
$|x_{i-1} - x_i| + |y_{i-1} - y_i| = 3, |x_{i-1} - x_i| \geq 1, |y_{i-1} - y_i| \geq 1$

On passe une fois et une seule sur chaque case sauf la première :
alldiff$(z_1, \ldots, z_{64})$ avec $z_i = 8(y_i - 1) + x_i, i = 1, \ldots, 64$

Type de problèmes abordés

problèmes *bien définis :*

- il existe une procédure qui permet de tester automatiquement toute solution proposée au problème et de la valider comme solution effective, si c'est le cas, en un nombre fini d'étapes.
- on doit être capable d'énumérer les solutions potentielles

ne sont pas considérés :

- les problèmes mathématiques résolus analytiquement ou numériquement
- les problèmes qu'on ne sait pas formaliser complètement

Méthodes de résolutions

Approches constructives

- procéder en complétant ou augmentant des solutions partielles
- décomposer le problème initial en sous-problèmes plus simples

exemples :

- n dames, taquin,
- tour de Hanoi, séparation/évaluation

CHAPITRE I

Recherche heuristique dans les graphes d'états

Plan

1. Formulation du problème
2. Algorithme A*
3. Terminaison et correction
4. Exemple
5. A* bi-directionnel

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 :

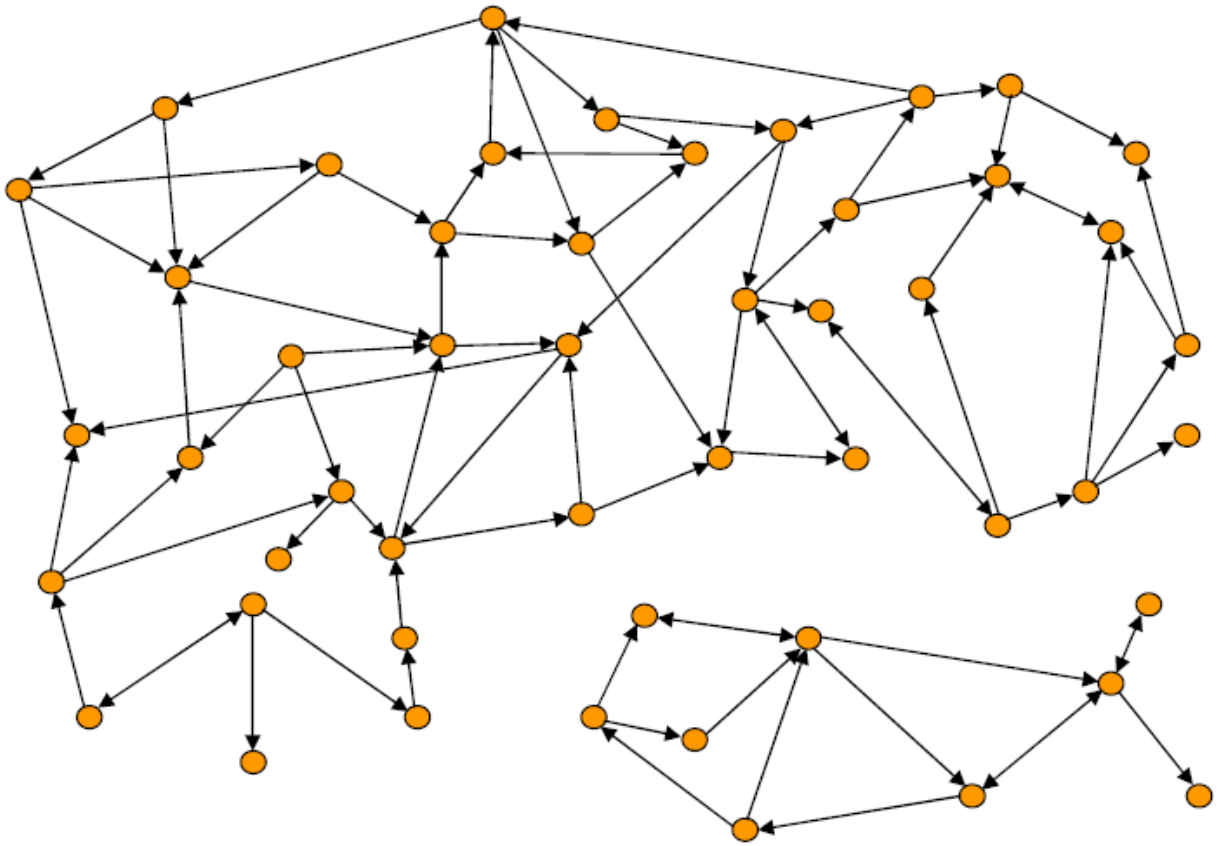
i) $\gamma \in \Gamma$

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

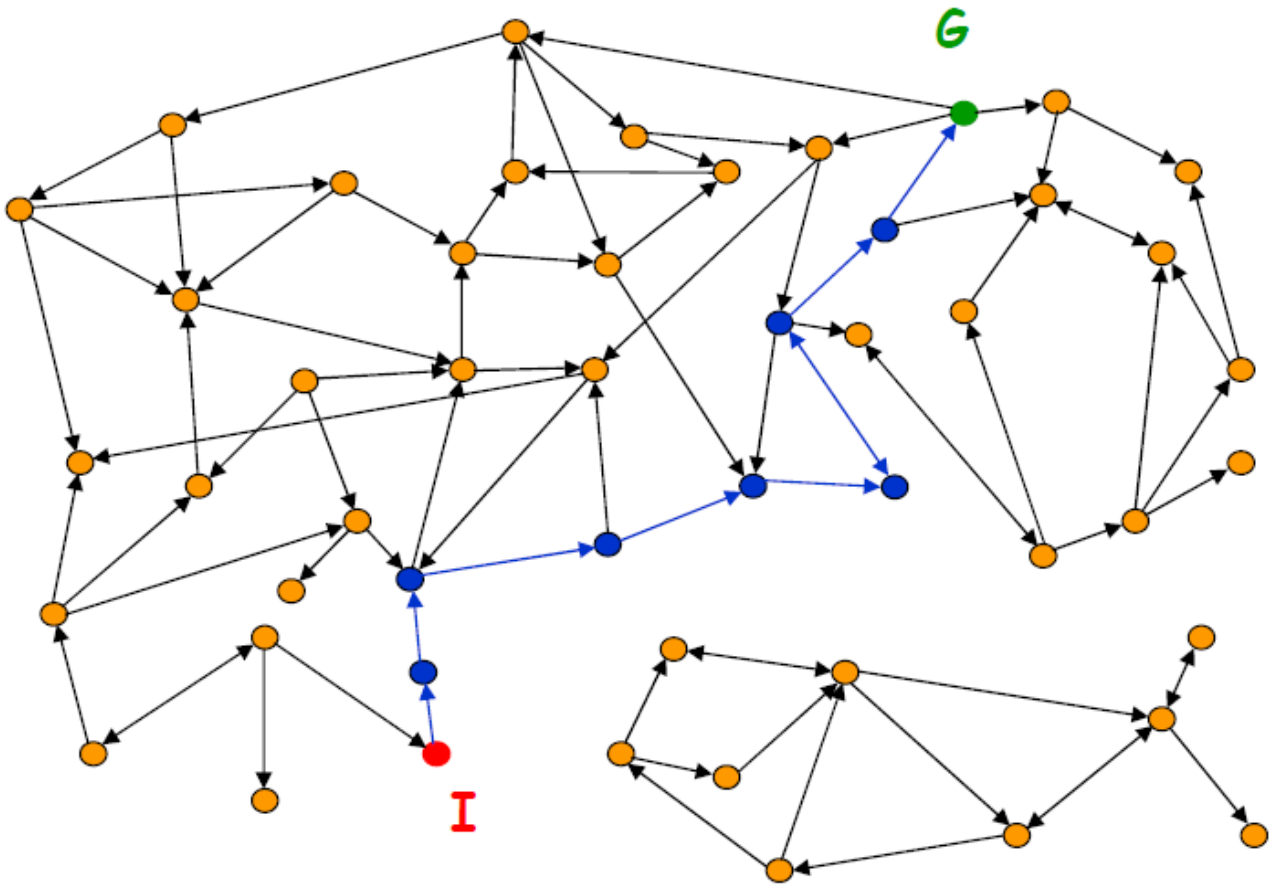
iii) $\sum_{i=1}^{r} k(n_{i-1}, n_i)$ est minimale sur l'ensemble des séquences vérifiant I) et II).

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.

Graphe d'états



Graphe d'états



Une procédure générale de recherche

PROCEDURE Recherche($G, n_0, \Gamma,$ 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 $f : N \to \mathbb{R}$

EXEMPLE 1 : *fonction profondeur* :

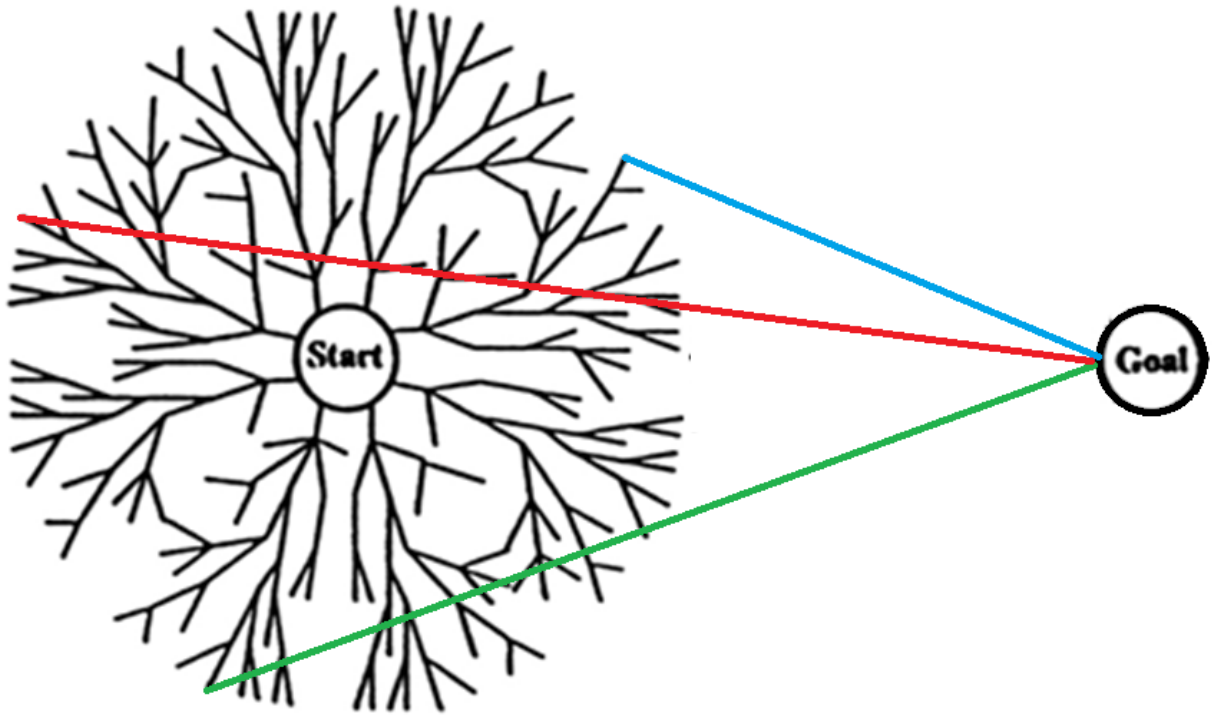
$f(n_0) = 0$
si $pere(n) = m$ alors $f(n) = f(m) + 1$

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

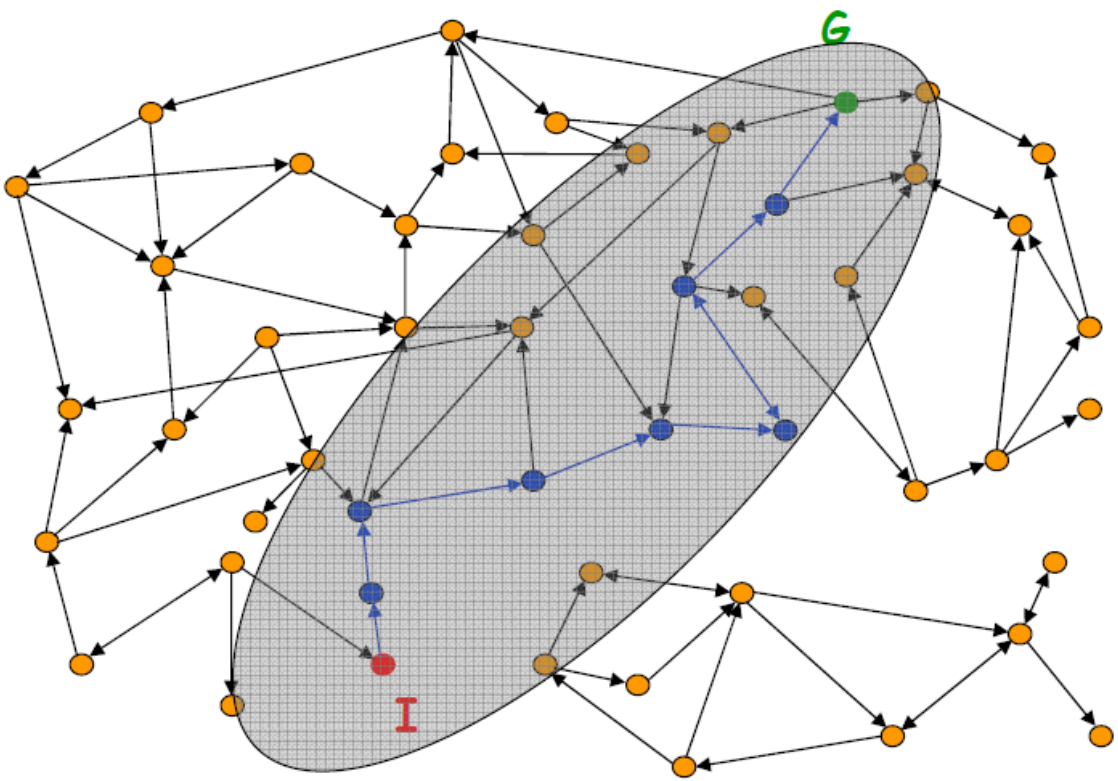
EXEMPLE 2 : *fonction d'évaluation* :

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

Recherche guidée par une heuristique



Recherche heuristique



Utilisation d'une heuristique

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 coîncidente est minorante

Preuve

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

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

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

□

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.

Illustration de A^*

Simulation de l'algorithme sur le problème du taquin :

- heuristique : nb de palets mal placés
- état intial :

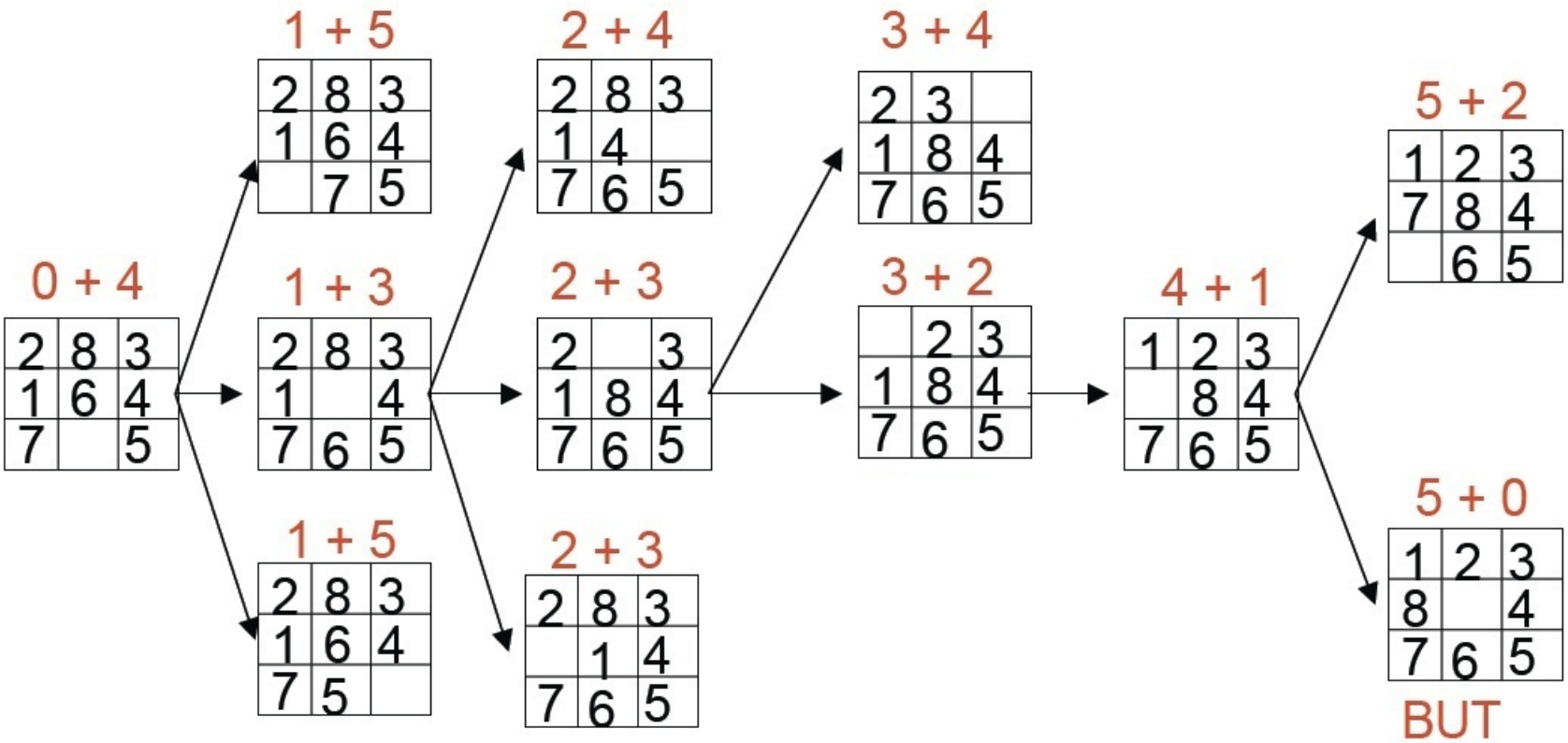
2	8	3
1	6	4
7		5

- état but :

1	2	3
8		4
7	6	5

- *autres heuristiques :* Manhattan, serpentin, combinaison

Arborescence développée par A^*



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 son coût et soit $P(k)$ l'ensemble des chemins solutions de coût inférieur ou égal à k. Comme le coût de tout chemin de $P(k)$ 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 $N' = \{n \in N, g^*(n) + h(n) \leq f^*(n_0) + M\}$. 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 optimal 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

□

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. On a alors un noeud but γ en tête des ouverts. D'après P4 il vérifie nécessairement : $f(\gamma) \leq f^*(n_0)$. Ce noeud est donc l'extremité terminale d'un chemin de coût minimal. □

En bref quelques autres propriétés importantes

Propriété 6 :
Si h est minorante alors la recherche se limite au sous graphe G' des noeuds de l'ensemble :

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

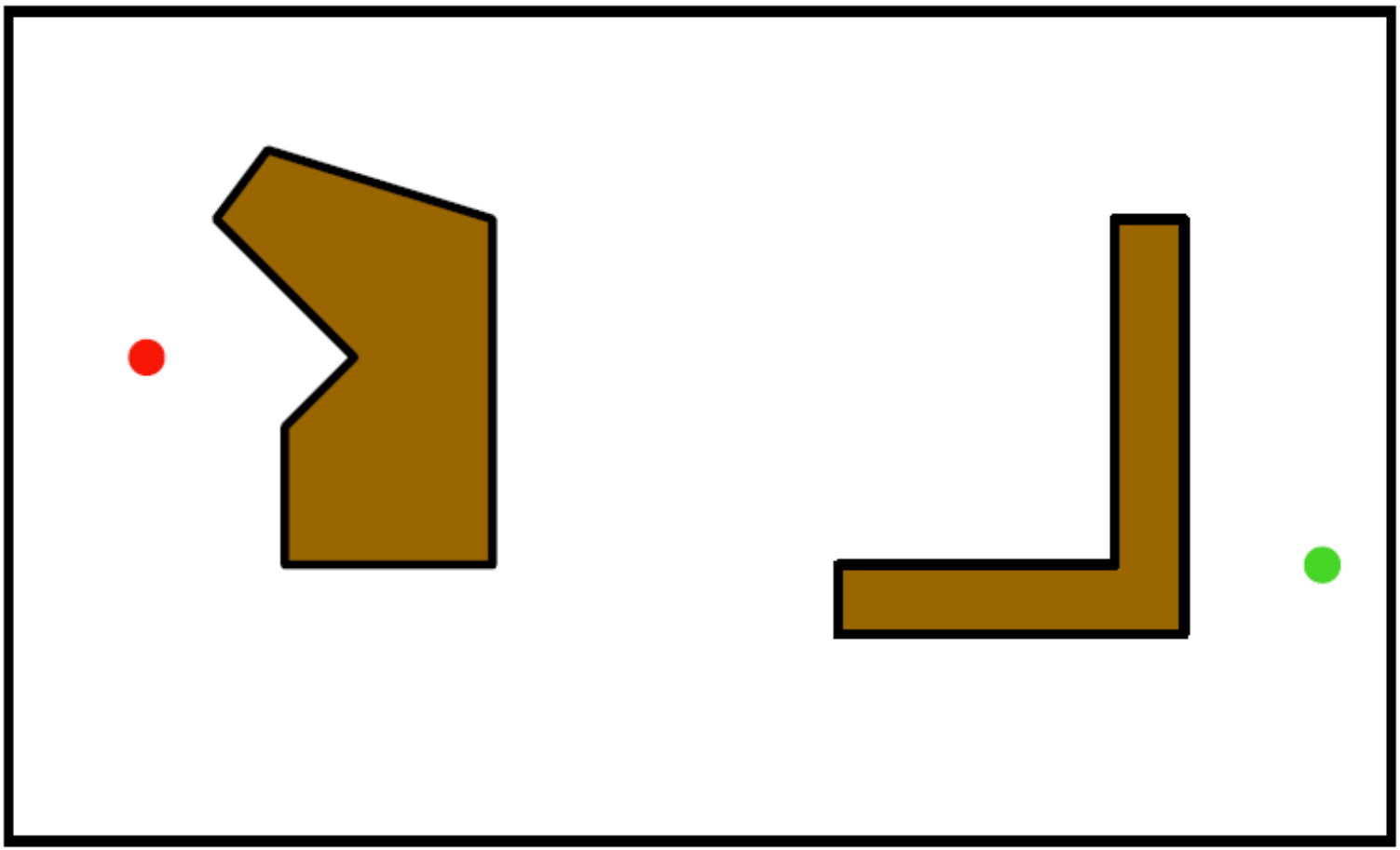
Propriété 7 : Si h est monotone alors tout noeud développé par A* est tel que $g(n) = g^*(n)$. Il s'ensuit qu'un noeud n ne peut être développé qu'au plus une fois.

Remarque : On montre 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})

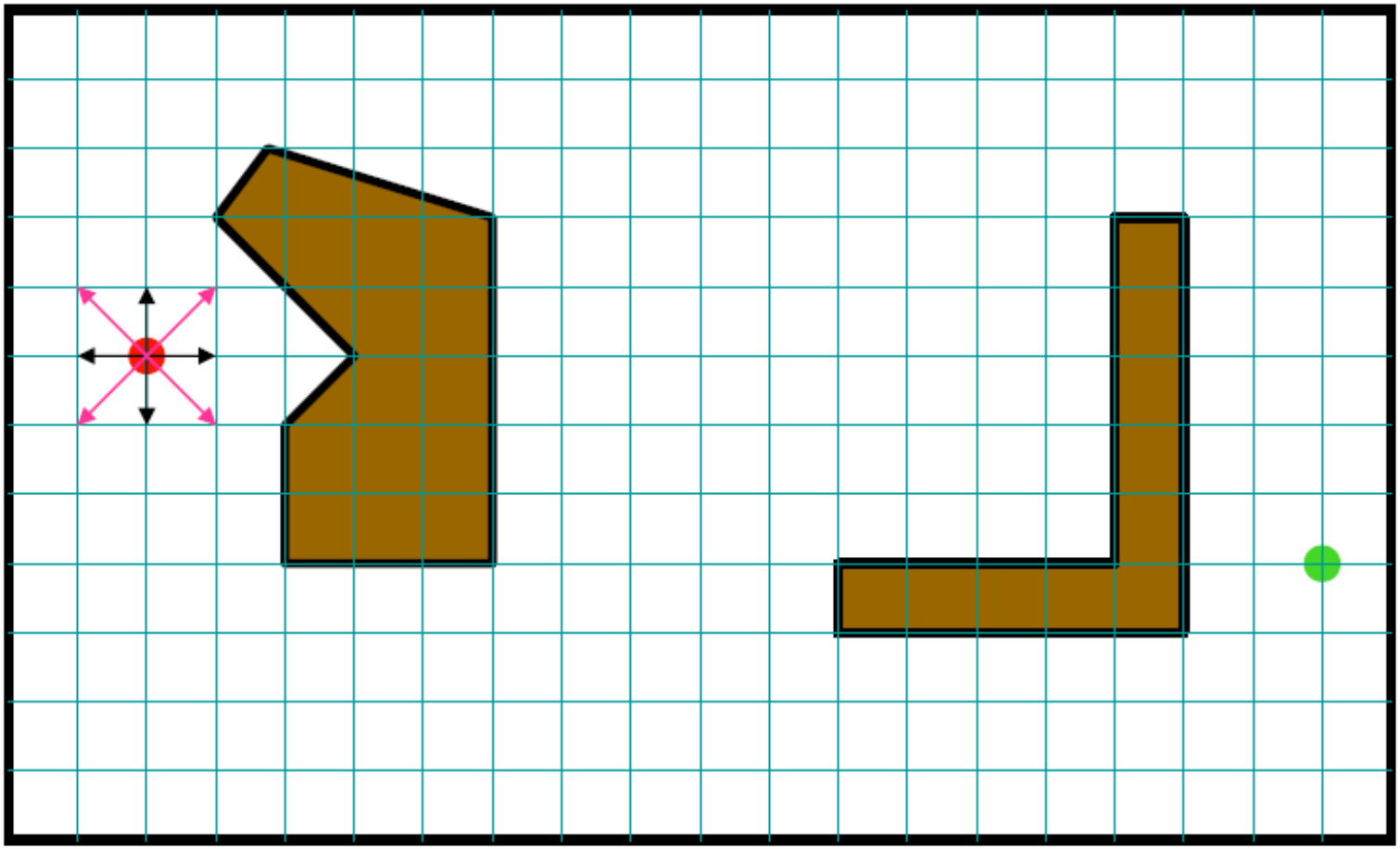
Application à la planification de trajectoires

Le problème initial



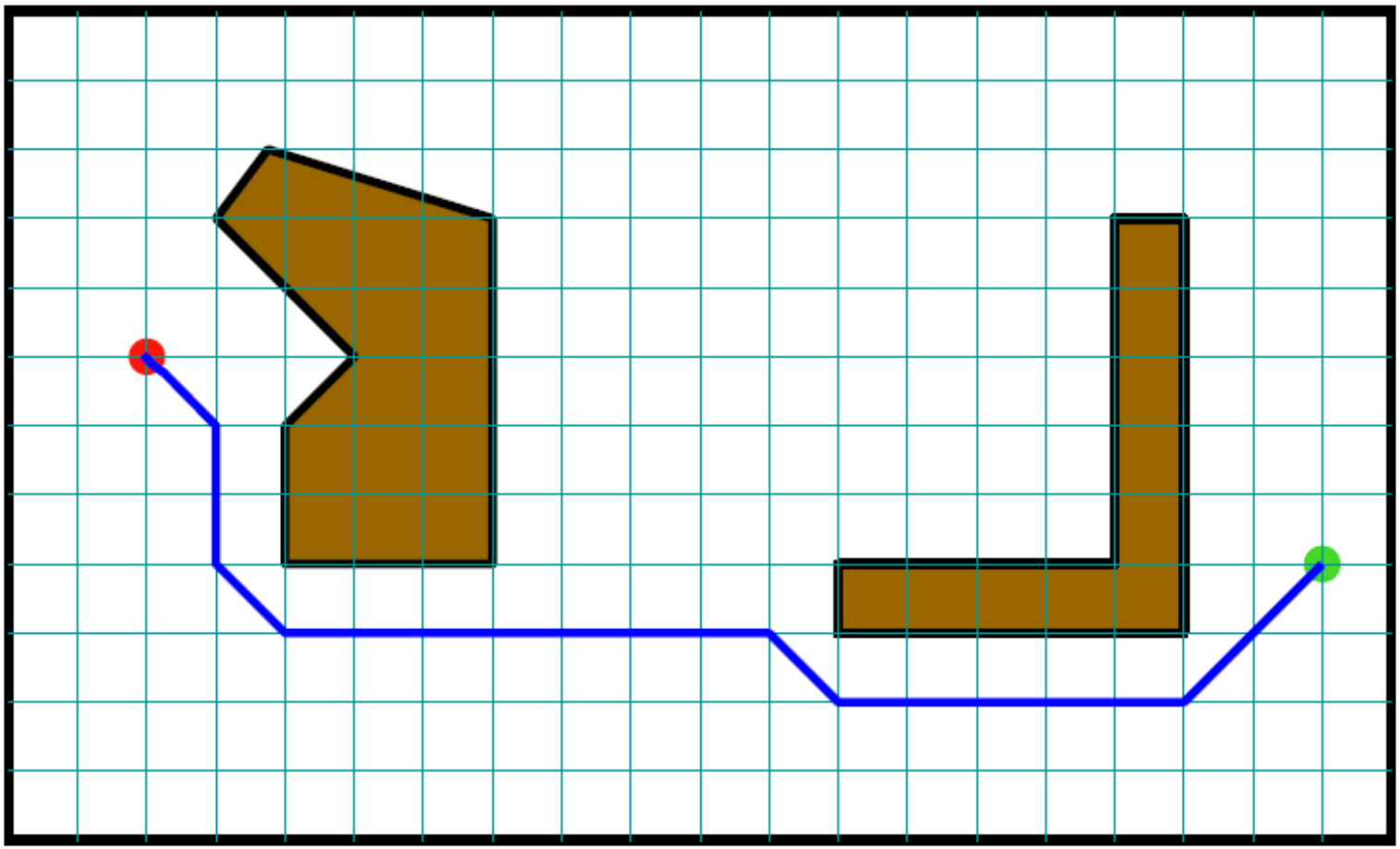
Application à la planification de trajectoires

Le problème discretisé :



Application à la planification de trajectoires

Le problème résolu



Exemple

Un robot dans un état initial e_1 doit atteindre l'état but e_7. Deux actions à sa diposition notées a_1 et a_2. Transitions :

Etat	actions	coût	état obtenu
e_1	a_1	3	e_2
	a_2	1	e_3
e_2	a_1	5	e_3
	a_2	1	e_4
e_3	a_1	4	e_4
	a_2	6	e_5
e_4	a_1	2	e_5
	a_2	4	e_7
e_5	a_1	3	e_6
	a_2	1	e_7
e_6	a_1	3	e_3
	a_2	1	e_7
e_7	–	–	–

Introduction d'une heuristique

Le robot dispose en chaque état d'une fonction heuristique h estimant le coût du chemin restant pour atteindre le but. La fonction h est la suivante :

e	e_1	e_2	e_3	e_4	e_5	e_6	e_7
$h(e)$	6	3	5	3	3	1	0

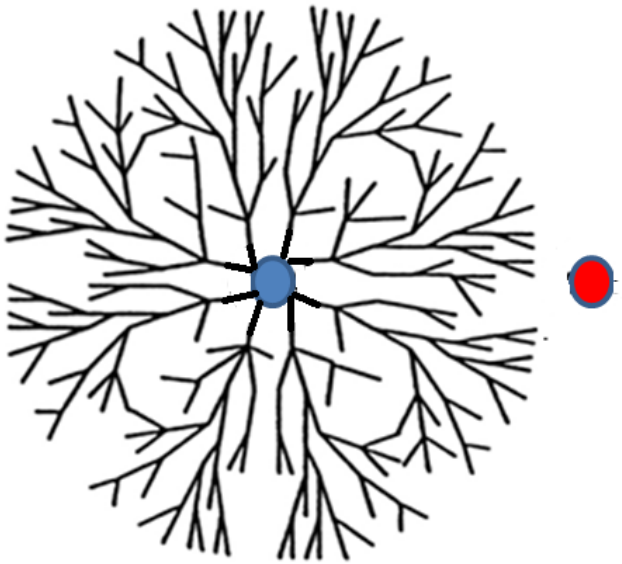
1) Déterminer la séquence d'actions retournée par l'algorithme A^* pour ce problème en utilisant l'heuristique h (on prendra soin de préciser la séquence d'états traversés et le coût du chemin associé).

Importance de l'heurisitique

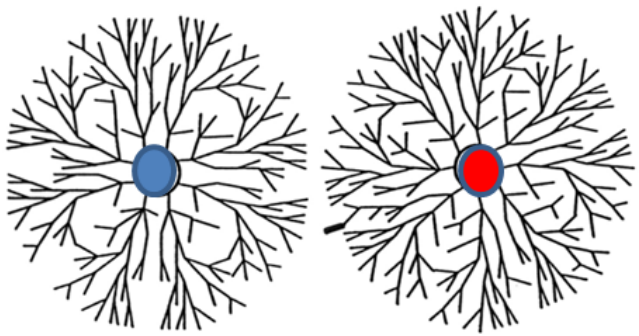
2) Montrer que l'heuristique utilisée n'est pas minorante.

3) Après avoir modifié cette heuristique de manière minimale pour la rendre minorante, redévelopper l'algorithme A^* avec cette nouvelle heuristique et déterminer le plan optimal du problème (on prendra soin de préciser la séquence optimale d'actions, la séquence d'états traversés et le coût du chemin associé).

Algorithme BA* (A^* bidirectionnel)



A* search



BA* search

Principe de BA^*

Déterminer le meilleur chemin de s à t en partant des 2 extrêmités

- on développe deux recherches alternées, une "en avant" partant de s l'autre "en arrière" partant de t
- on utilise deux ensembles d'ouverts un à gauche OG et un à droite OD et deux ensembles fermés FG et FD
- on utilise deux fonctions g et deux fonctions h :
  - $g_s(n)$ coût du meilleur chemin trouvé de s à n
  - $g_t(n)$ coût du meilleur chemin trouvé de t à n
  - $h_t(n)$ estimation de la distance de n à t
  - $h_s(n)$ estimation de la distance de n à s
- un élément n de OG s'évalue par $f_g(n) = g_s(n) + h_t(n)$
- un élément n de OD s'évalue par $f_d(n) = g_t(n) + h_s(n)$

Développement et test d'arrêt

on note μ la valeur du meilleur chemin déjà trouvé (initialement $\mu = \infty$)

On répète les développements alternés suivants :

choisir un noeud $n \in OG$ minimisant $f_g(n)$ et le développer

choisir un noeud $n \in OD$ minimisant $f_d(n)$ et le développer

Si les deux frontières d'ouverts OG et OD se rejoignent à un noeud n alors $g_s(n) + g_t(n)$ peut être utilisé pour mettre à jour μ si $g_s(n) + g_t(n) < \mu$

Condition d'arrêt :

$$\mu \leq \max\{\min_{n \in OG} f_g(n), \min_{n \in OD} f_d(n)\}$$

Deux principes complémentaires :

- nipping : si on selectionne un noeud dans OG pour le développer et qu'il est déjà dans FD on peut ne pas le développer et le mettre dans FG ;
- pruning : dans la même situation, les descendants de ce noeud présents dans OD peuvent être supprimés.

Application à l'exemple

Dans l'exemple précédent on considère les fonctions heuristiques gauches et droites suivantes :

e	e_1	e_2	e_3	e_4	e_5	e_6	e_7
$h_g(e)$	6	3	5	3	1	1	0
$h_d(e)$	0	3	1	3	5	5	6

Appliquer l'algorithme BA* à ce problème.