EFREI 2011/2012
L3

Aide à la Décision
Recherche dans un espace d'états – Recherches heuristiques

Support de cours

Recherche heuristique

Heuristic search

 3

Stratégies de recherche

- Recherche aveugle :
  - Aucune information sur la structure de l'arbre de recherche
  - « rustique » / systématique
- Recherche heuristique :
  - Information disponible
  - Amélioration du processus de recherche

 4

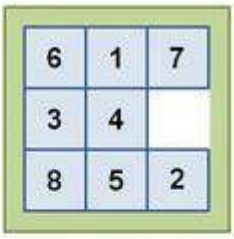
Fonction heuristique

- $h : Q \rightarrow \mathcal{R}$
  - Associe une valeur à chaque état
    - La valeur de h(e) est indépendante des circonstances dans lesquelles on « rencontre » l'état 'e'
- h(e) = estimation du rapport coût / bénéfice en passant par 'e' pour atteindre une solution
  $\approx C^*(e,\ldots,q_s)$ … *ou* $fct(C^*(e,\ldots,q_s))$
  avec q_s un état solution
- Propriété fondamentale (mais parfois théorique…) :
  h (état solution) = 0

 5

Taquin (« 8-puzzle »)

- Nombre de pièces mal placées
  - 6, 1, 7, 3, 4, trou, 8, 5, 2
    soit 9 pièces mal placées

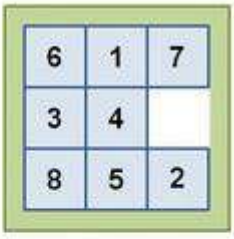


Etat solution
h = 0

(C) Hervé Barbot, 2005-2011 6

Taquin (« 8-puzzle »)

- Distance Manhattan
  - Pour chaque pièce, nombre de déplacements pour l'amener à sa position finale :
    - 1 : 1 2 : 3 3 : 3
      4 : 1 5 : 1 6 : 3
      7 : 4 8 : 1 0 : 1
      h = ∑ = 18



(C) Hervé Barbot, 2005-2011 7

h : Q → ℛ
h(e) = estimation du rapport coût / bénéfice
en passant par 'e' pour atteindre une solution

- Plus une valeur h(e) est petite, plus l'état 'e' est supposé « proche » d'un état solution
  - Il y a donc probablement tout bénéfice à l'explorer avant les autres !

(C) Hervé Barbot, 2005-2011 8

- Soit $h_{réel}$ (e) le coût réel minimum de 'e' à un état solution
  - $h_{réel}$ est une valeur qui n'est pas calculée !
  - $h_{réel}$ est inconnue !!

- h (e) – $h_{réel}$ (e) évalue la qualité de la fonction heuristique h choisie
  - La performance d'un algorithme de recherche heuristique dépendra de la fonction h choisie !

(C) Hervé Barbot, 2005-2011 9

- Fonction « monotone » / « consistante » :

$$h(e) - h(s) < C(e,s)$$

avec $e,s \in Q$, $s \in S(e)$

- Fonction « presque parfaite » :

$$h_{réel}(q) > h_{réel}(q') \Rightarrow h(q) > h(q')$$

h établit une relation d'ordre identique à celle définie par $h_{réel}$

L'ordre de choix des états dans l'ensemble 'ouverts' sera identique à celui correspondant à une mise en œuvre théorique de $h_{réel}$!

(C) Hervé Barbot, 2005-2011 10

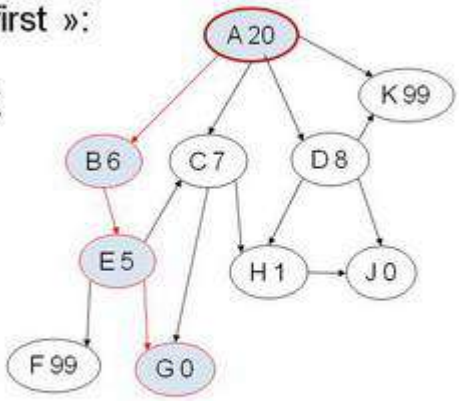
Principe d'une recherche « best-first »

- Principe : combine profondeur et largeur d'abord
  - en profondeur : solution trouvée sans avoir besoin de calculer tous les nœuds
  - en largeur : ne risque pas de rester pris dans un cul-de-sac

- Analyse des nœuds dans l'ordre (croissant / décroissant) de leurs valeurs associées

(C) Hervé Barbot, 2005-2011 11

- Une approche « best-first »:
  (1) Etat initial : A
  Successeurs : B, C, D, K
  (3) Choix de B
  Successeurs : E
  (4) Choix de E
  Successeurs : F, G
  (5) Choix de G
  ARRET si solution



Cette solution (ABEG) est-elle meilleure que ACG, voire ACHJ, ADJ, ou ABECG ?

(C) Hervé Barbot, 2005-2011 12

- Introduction
- Description formelle d'un problème de recherche
- Exploration d'arbres
- Recherches aveugles
  - Principes généraux
  - Illustration
  - Recherche en profondeur d'abord
  - Recherche en profondeur limitée
  - Approfondissement itératif
  - Recherche en largeur d'abord
  - Recherche en coût uniforme
- Recherches heuristiques
  - Principes généraux
  - Recherche gloutonne / recherche gourmande
  - A*
- Propriétés des méthodes de recherche – Comparaison
- Conclusion & ouverture

(C) Hervé Barbot, 2005-2011 13

Recherche gloutonne / gourmande

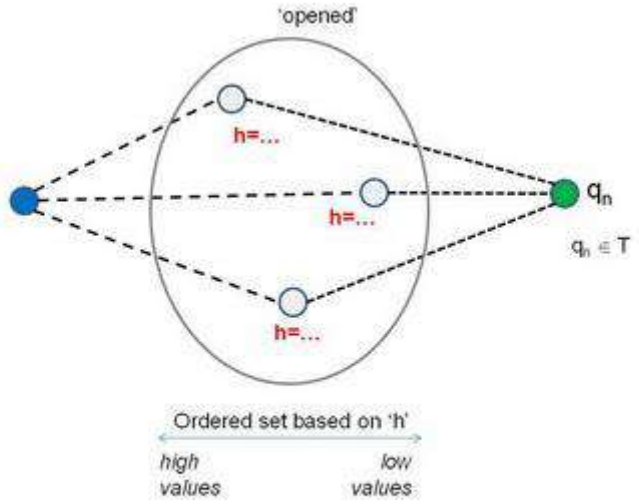
« greedy search »

- Principe : à chaque étape, choisir un état ayant le « coût restant estimé » minimum

Parmi l'ensemble des états candidats à expansion s_1, s_2, …, s_n, on prend un état s tel que

$$h(s) = MIN(h(s_i))$$

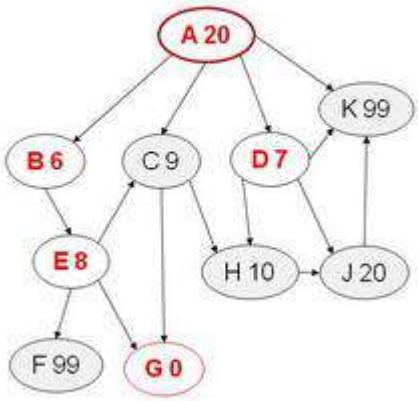
(C) Hervé Barbot, 2005-2011 14



Greedy search strategy: select the lowest values of 'h'

(C) Hervé Barbot, 2005-2011 15

Closed	Opened
	A
A	B C D K
A B	C D K E
A B D	C K E H J
A B D E	C K H J F G
A B D E G	



(C) Hervé Barbot, 2005-2011 16

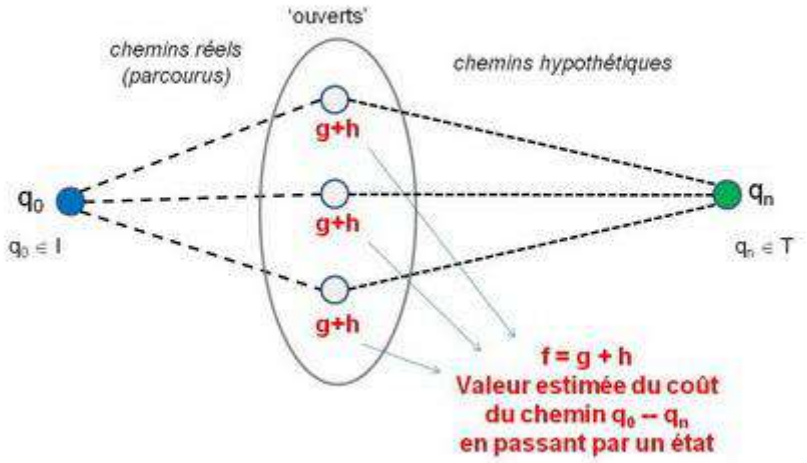
```
opened ← { initial state }
closed ← ∅
succes ← false

Loop while opened ≠ ∅ ∧ ¬ succes do

  e ← select_minimum_h_in ( opened )

  is_final(e)    ⇒ ...
  ¬ is_final(e)  ⇒ ...
```

(C) Hervé Barbot, 2005-2011 17

- Introduction
- Description formelle d'un problème de recherche
- Exploration d'arbres
- Recherches aveugles
  - Principes généraux
  - Illustration
  - Recherche en profondeur d'abord
  - Recherche en profondeur limitée
  - Approfondissement itératif
  - Recherche en largeur d'abord
  - Recherche en coût uniforme
- Recherches heuristiques
  - Principes généraux
  - Recherche gloutonne / recherche gourmande
  - A*
- Propriétés des méthodes de recherche – Comparaison
- Conclusion & ouverture

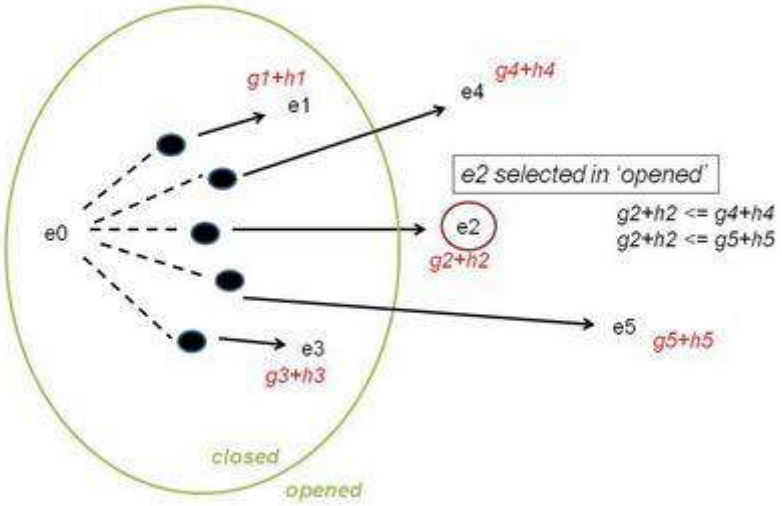
(C) Hervé Barbot, 2005-2011 18

A *

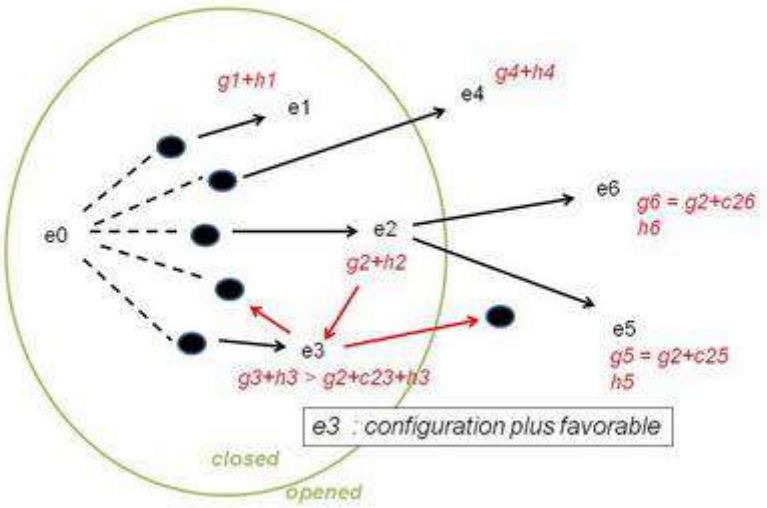


(C) Hervé Barbot, 2005-2011 19

Exemple



(C) Hervé Barbot, 2005-2011 20



(C) Hervé Barbot, 2005-2011 21

Remise en cause

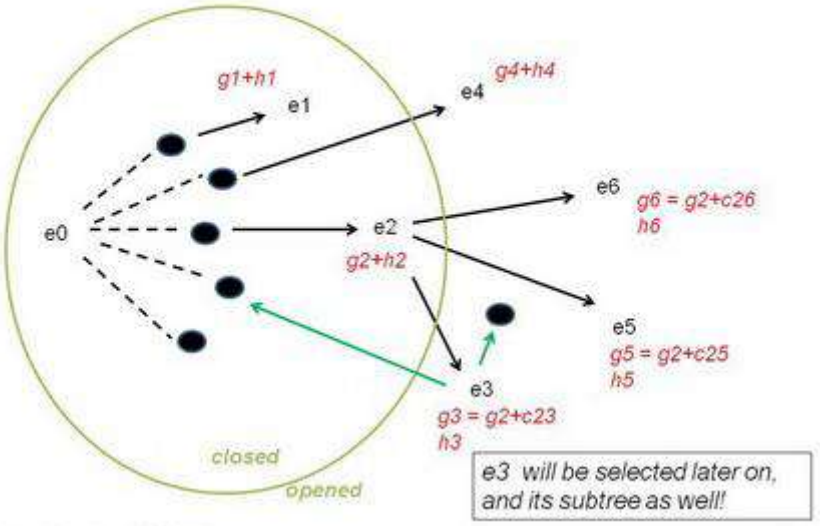
- Problème : que se passe-t-il pour les descendants 'x' (successeurs directs et indirects) de e3 ?
  - g3+h3 > g2 + c23 + h3
    g3 > g2 + c23
    donc g3 va diminuer !
  - gx = g3 + « quelque chose » $x \in S(e3)$
    gx doit logiquement diminuer

(C) Hervé Barbot, 2005-2011 22

Remise en cause

- Solutions :
  - On analyse immédiatement toute la descendance de e3
    - Mais comment faire si on a déjà exploré un sous-arbre relativement important « sous » e3 ?
      - D'autant plus que la seule information liée à la structure de l'arbre est la fonction « prédécesseur »…
  - On laisse A* fonctionner tout seul pour le faire

(C) Hervé Barbot, 2005-2011 23



 24

```
¬ is_final(e) ⇒ opened ← opened - e
                closed ← closed + e
                ∀ x ∈ S(e),
                   x ∉ opened ∪ closed
                       ⇒ opened ← opened+{x}
                          g(x) ← g(e)+C(e,s)
                          S⁻¹(x) ← e
                   x ∈ opened ∧ f(x) > g(e)+C(e,x)+h(x)
                       ⇒ g(x) ← g(e)+C(e,s)
                          S⁻¹(x) ← e
                   x ∈ closed ∧ f(x) > g(e)+C(e,x)+h(x)
                       ⇒ g(x) ← g(e)+C(e,s)
                          S⁻¹(x) ← e
                          closed ← closed - {x}
                          opened ← opened - {x}
```

 25