# Algorithmes de recherche locale

IA – Chapitre 4b

Master STIC - Université d'Orléans - Avril 2009

# Aperçu

- Recherche par escalade (Hill-climbing)
- Recuit simulé (Simulated annealing)
- Local beam search
- Recherche locale dans des espaces continus (brièvement)
- Algorithmes génétiques (brièvement)

# Algorithmes de recherche locale

- Les algorithmes de recherche aveugles et A* explorent l'espace de recherche systématiquement et mémorisent le chemin vers le nœud actuel.
  $\Rightarrow$ Arrivant au nœud final, le chemin constitue une solution.
- Dans plusieurs problèmes, le chemin n'est pas pertinent, par exemple le problème de 8 reines.
- Les algorithmes de recherche locale ne se soucient pas du chemin emprunté
  $\Rightarrow$ adaptés aux problèmes dont on ne cherche qu'une solution et non un chemin pour y arriver
- Les algorithmes de recherche locale considèrent un état courant puis se déplacent successivement vers un état voisin par une relation de voisinage.

# Propriétés des algorithmes de recherche locale

- Ils sont
  – non-complets
  – non-optimaux
  – non-exhaustifs
- Mais
  – utilisant très peu de mémoire, souvent en quantité constante, ils conviennent aux recherches “offline” et “online”
  – ils trouvent souvent des solutions raisonnables dans des espace d’états très grands voir infinis, où les autres algorithmes ne peuvent pas s’exécuter
  – ils s’adaptent aux problèmes d’optimisation, où l’on cherche le meilleur état suivant une fonction d’objectif.

# Voisinage

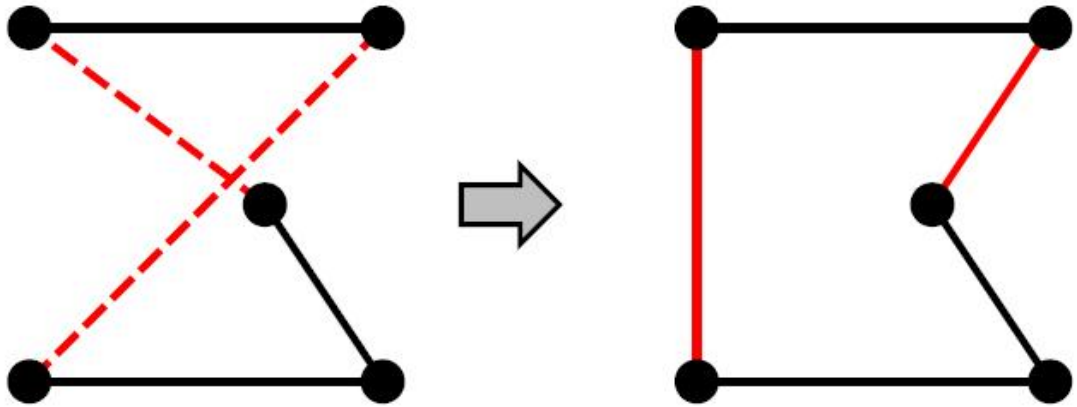
- Un voisinage s'obtient en perturbant légèrement un état
- On définit une distance

$$V(s) = \{s' \mid d(s, s') < \epsilon\}$$

- On définit une fonction de coût qui mesure l'écart entre l'état et l'objectif ou simplement la valeur de l'état courant $\Rightarrow$ optimisation de cette fonction

# Exemple : Le voyageur de commerce

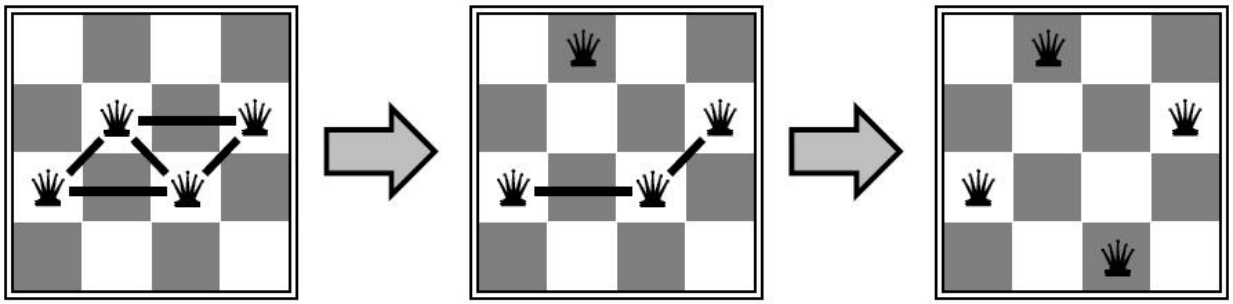
L'état initial est un tour complet quelconque, puis les états voisins sont obtenus successivement en échangeant deux déplacements.



Des variantes de cette méthodes trouvent une solution raisonnable rapidement avec de centaines de villes.

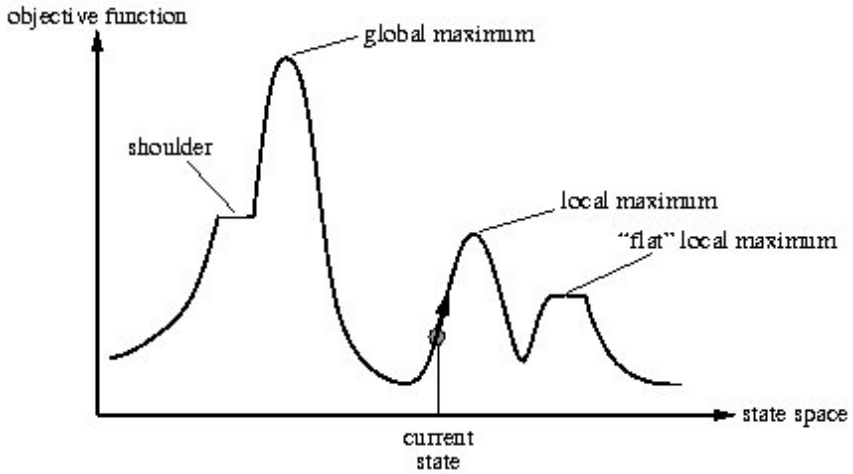
# Exemple : $n$-reines

Placer $n$ reines sur un échiquier $n \times n$ de telle façon que deux reines ne se retrouvent sur une même ligne, même colonne, ou même diagonale
Déplacer une reine pour réduire le nombre de conflits




Trouver souvent une solution presque instantanément pour de très grand $n$, par exemple $n = 1$ million

# Le paysage d'un espace d'états




Les algorithmes de recherche locale explorent ce paysage.
Rappel :

- Algorithme complet : toujours trouver une solution s'il en existe une
- Algorithme optimal : toujours trouver le maximum/minimum global

# Recherche par escalade

- Le principe consiste à rechercher, parmis les voisins, celui qui améliore le plus la fonction de coût
- On recommence jusqu'à obtenir un maximum
  $\Rightarrow$ Il s'agit très souvent d'un maximum local
  $m$ est maximum local si

$$\forall m' \in V(m) \;\; f(m') < f(m)$$

- L'algorithme est glouton

# Algorithme de recherche par escalade

```
function HILL-CLIMBING(problem) returns a state that is a local maximum
   inputs: problem, a problem
   local variables: current, a node
                    neighbor, a node

  current ← MAKE-NODE(INITIAL-STATE[problem])
   loop do
      neighbor ← a highest-valued successor of current
       if VALUE[neighbor] ≤ VALUE[current] then return STATE[current]
      current ← neighbor
   end
```

# Exemple : SAT

Problème de satisfaction d'une formule booléenne, mise sous forme de clauses

$$\bigwedge_{i \in I} (a_1 \vee a_2 \vee \cdots \vee a_{k_i})$$

où $a_j$ est une variable $x$ ou la négation d'une variable $\neg x$

- Variables $V = \{x_1, \ldots, x_n\}$
- Etat = affectation de $n$ variable booléennes
  $2^n$ états, problème NP-Complet

# Exemple : SAT (voisinage)

- Voisin : état qui diffère par 1 bit
  $\longrightarrow$ distance de Hamming = 1 (nombre de bits différents)
  Exemple

| $x_1$ | $x_2$ | $x_3$ | $x_4$ | $x_5$ | |
|---|---|---|---|---|---|
| 0 | 1 | 1 | 0 | 1 | un état |
| 1 | 1 | 1 | 0 | 1 | voisins à distance 1 |
| 0 | 0 | 1 | 0 | 1 | |
| ⋮ | | | | ⋮ | |
| 1 | 0 | 1 | 0 | 1 | voisins à distance 2 |
| 1 | 1 | 0 | 0 | 1 | |
| ⋮ | | | | ⋮ | |

# Exemple : SAT (fonction d'évaluation)

- Fonction d'évaluation d'un état

$$f(s) = \begin{cases} 0 & \text{si la formule s'évalue à faux} \\ 1 & \text{sinon} \end{cases}$$

  Problème : cette fonction ne nous informe pas sur la distance à la solution. Que l'on ait 1 bit à changer ou 1000 bits pour arriver à une solution, $f$ vaut 0.

- Fonction $f'(s)$ = nombre de clauses satisfaites. S'il y a $m$ clauses, $f'(s) = m$ signifie que $s$ est une solution
  Problème : satisfaire une clause courte a autant de mérite qu'une grande

- Fonction

$$f(s) = \frac{\text{nombre de clauses satisfaites} \times \text{nombre de variables}}{\text{nombre de variables total}}$$

# Exemple : Algorithme GSAT

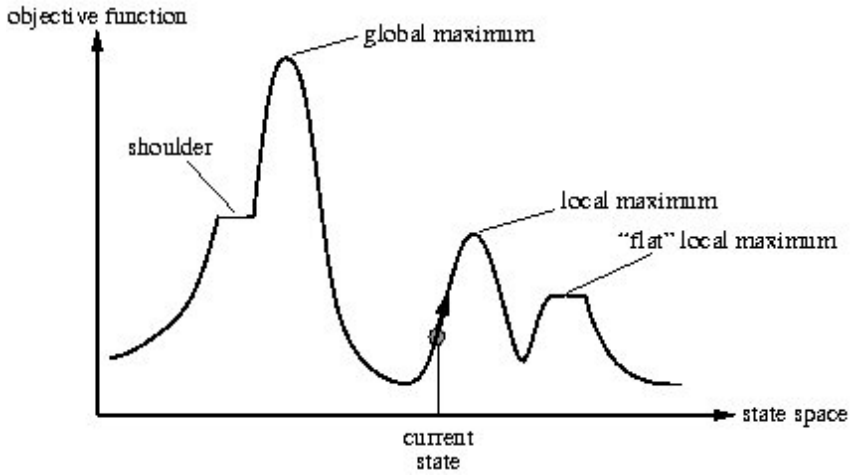
```
function GSAT( SAT problem)  returns a solution state
   inputs: SAT problem, a problem
   local variables: current, a node
                    neighbors, a list of nodes
                    v, a node

  current ← a random state
   for i=1 to  MAX-FLIP do
       if current(formula)=true  then return current
      neighbors ← exchange the value of one variable
      v ← a neighbor having the highest-value f
      current ← v
    end
```

# Exemple : Algorithme GSAT (2)

- L'algorithme peut osciller dans un maximum local
- S'il n'y a pas de réponse, on peut relancer l'algorithme
- Il existe beaucoup de versions de cet algorithme
- L'algorithme a été inventé par Selman (1992)

# Améliorer la technique d'escalade




Il faut pouvoir sortir des maxima locaux

Random-restart hill climbing

Random sideways moves :-) sort des épaulements :-( risque de boucle sur un plateau

# Le recuit simulé

Idée : autoriser de redescendre vers un état moins bon avec une certaine probabilité

- La probabilité de descente dépend d'un paramètre appelé température
- Au cours de la recherche, on diminue progressivement la température
- Température élevée : recherche + aléatoire
  $\longrightarrow$ exploration de larges zones de l'espace de recherche
- Recuit simulé = simulated annealing

# Algorithme du recuit simulé

On utilise une fonction temps qui associe une température à un numéro d'itération

```
function SIMULATED-ANNEALING( problem, schedule)  returns a solution state
   inputs: problem, a problem
           schedule, a mapping from time to "temperature"
   local variables: current, a node
                    next, a node
                    T, a "temperature" controlling prob. of downward steps

  current ← MAKE-NODE(INITIAL-STATE[problem])
  for t ← 1 to ∞ do
      T ← schedule[t]
      if T = 0 then return current
      next ← a randomly selected successor of current
      ΔE ← VALUE[next] – VALUE[current]
      if ΔE > 0 then current ← next
      else current ← next only with probability e^(Δ E / T)
```

# Influence de la température

- Température $T$ élevée : on garde une forte probabilité d'accepter un mouvement assez mauvais
- Température $T$ faible : la probabilité d'accepter un mouvement mauvais est très faible
- Si la baisse de température est suffisament progressive, l'optimal global est trouvé

# Local beam search

Idée: garder $k$ états au lieu d'un ; choisir les $k$ meilleurs états parmis les successeurs
Pas le même comportement avec $k$ recherches en parallèle !
Les recherches qui trouvent des meilleurs états appellent les autres à les joindre
Problème: assez souvent $k$ états finissent par se retrouver dans un même maximum local
Idée : choisir $k$ successeurs aléatoirement, stochastic beam search
Analogie dans la sélection naturelle !

# Exploration d'espaces continus

- Supposons qu'on veuille situer 3 aéroports en Roumanie
  - problème à 6 dimensions réelles :
    $(x_1, y_2)$, $(x_2, y_2)$, $(x_3, y_3)$
  - fonction objectif (exemple) :
    $f(x_1, y_2, x_2, y_2, x_3, y_3)$
    = somme des distances (au carré) de chaque ville au plus proche aéroport
  - Comment procéder ?
- Approche générale : dérivées partielles
- Discrétisation
- Gradient

# Discrétisation

- Transformation de l'espace continu en espace discret
  e.g., gradient empirique
  - on fixe un pas $\delta$
  - on mesure la fonction objectif pour un pas $\pm\delta$ dans chaque direction
  - on se déplace dans la direction apportant le plus fort gain
  - problème : valeur de $\delta$...
  - une technique possible : exploration linéaire
    - la valeur de $\delta$ est augmentée (e.g. doublée) tant que f croît
    - le point atteint avec $\delta_{max}$ est le nouvel état courant

# Méthode du gradient

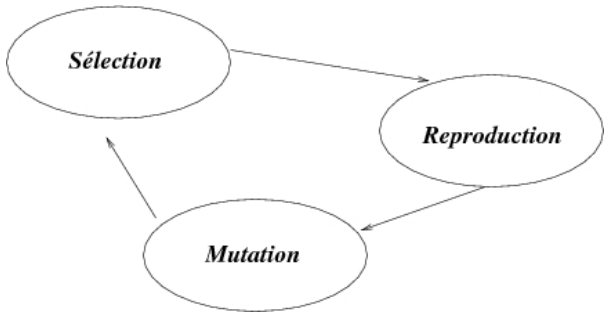
- La méthode du Gradient calcule

$$\nabla f = \left( \frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial y_1}, \frac{\partial f}{\partial x_2}, \frac{\partial f}{\partial y_2}, \frac{\partial f}{\partial x_3}, \frac{\partial f}{\partial y_3} \right)$$

  l'état peut alors être mis à jour en suivant la plus forte pente, en utilisant $x \leftarrow x + \alpha \nabla f(x)$ ($\alpha$ peut être géré avec l'exploration linéaire).
- Notons qu'un maximum est atteint quand $\nabla f(x) = 0$.
- Pour rechercher ce point, on peut avoir recours à la méthode de Newton–Raphson, qui itère
  $x \leftarrow x - H_f^{-1}(x) \nabla f(x)$
  $H_f$ est la matrice hessienne des dérivées secondes
  $H_{ij} = \partial^2 f / \partial x_i \partial x_j$

# Les algorithmes génétiques

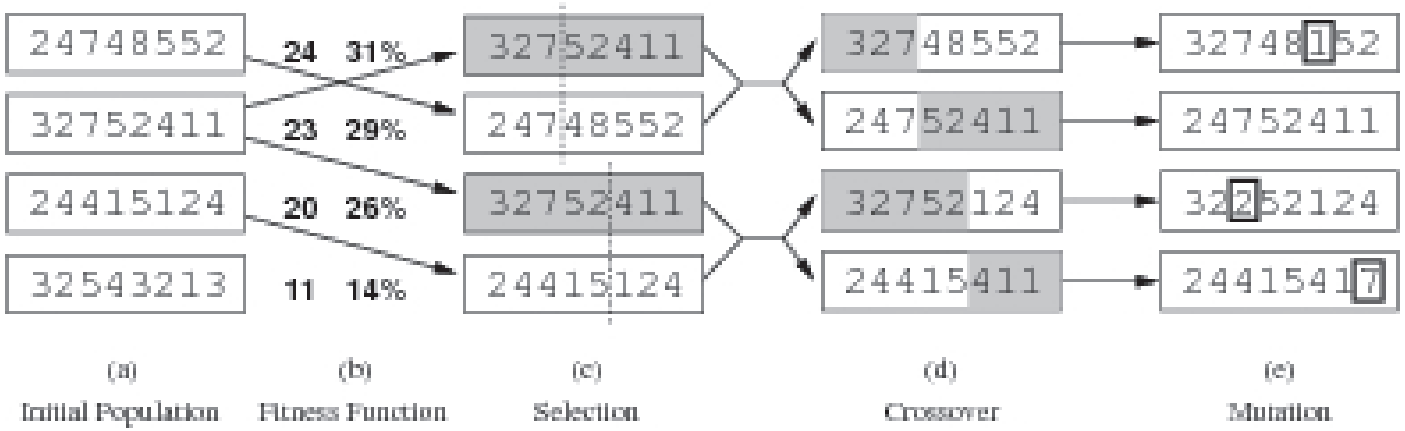
- Le principe
  – maintenir une population de solutions candidates appelées individus (= états)
  – coder chaque individu : ses gènes
  – générer au hasard une population initiale
  – exécuter le cycle suivant jusqu'à obtention d'un critère d'arrêt




- Le pari : le croisement de 2 bons individus a de fortes chances de produire un individu meilleur

# Algorithmes génétiques

Algorithmes génétiques = “stochastic local beam search” + generation des successeurs à partir de *pairs* d’états



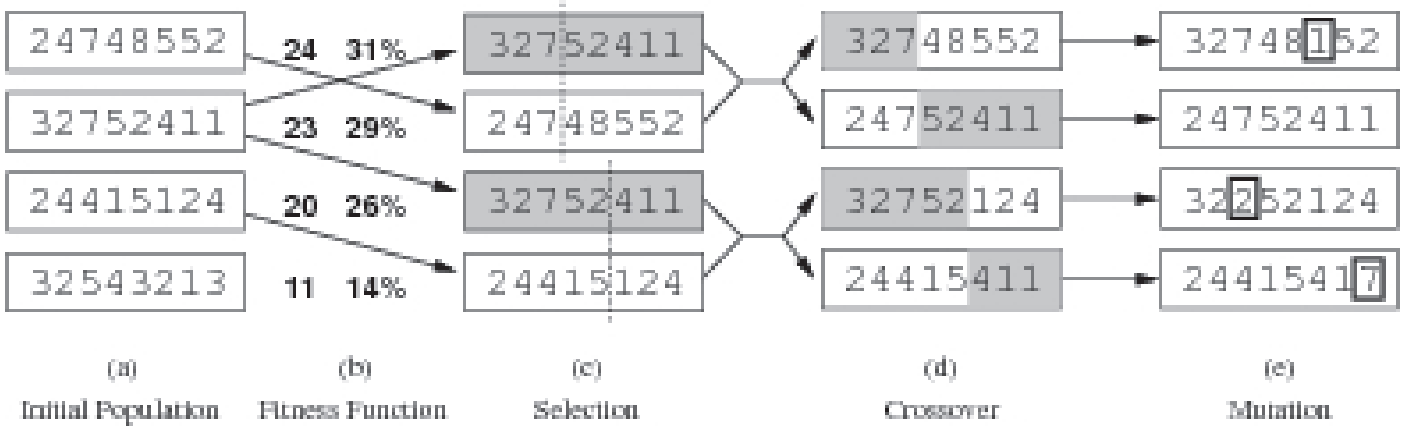

Les algorithmes génétiques utilisent les états codés sous forme de chaînes

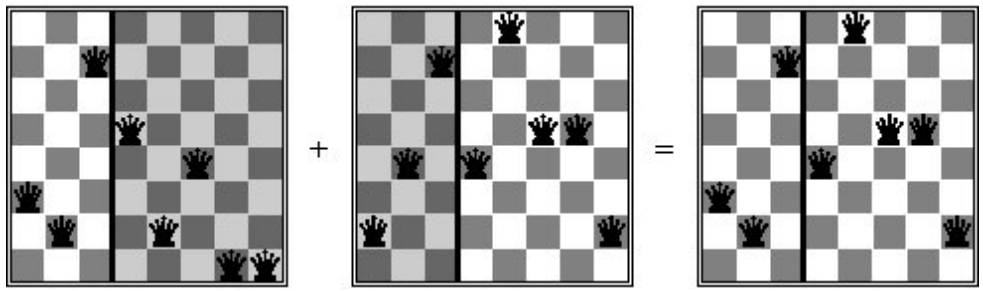
Le croisement a une signification si des sous-chaînes ont une signification

# Evaluation et reproduction des individus

- Fonction d'adaptation (fitness)
- hybridation (recombinaison)
  Les paires d'individus (chromosomes) sont recombinées à un certain point (choisi aléatoirement).
  Faut-il (peut-on) couper n'importe où ?
  - notion de schéma / d'instance de schéma
  - importance de la modélisation (codage)
- mutation
  La valeur de certains emplacements est modifiée de façon aléatoire.

# Exemple : 8 reines



Fonction d'adaptation ?

# Méthodes de sélection (1)

- Sélection par roulette
  - La probabilité de sélection des parents est proportionnelle à leur performance.
  - Pb : risque de surreprésentation.
- Sélection par rang
  - Les chromosomes sont triés par performance. La probabilité dépend du rang des chromosomes (et non plus directement de leur performance).
  - Meilleure représentativité, risque de convergence plus lente.
- Sélection par tournoi
  - $m$ chromosomes $\longrightarrow$ $m$ paires; on fixe la probabilité de victoire du plus fort (typiquement : de 70 à 80 %; $\longrightarrow$ $m$ chromosomes pour la reproduction.

# Méthodes de sélection (2)

- Sélection steady-state
  - Approche différente : une grande partie de la population est conservée. On remplace les plus mauvais chromosomes par de nouveaux, produits à partir des meilleurs.
  - Intérêt ?
- Elitisme
  - Pour éviter de perdre les meilleurs chromosomes lors de la reproduction, on les reproduit à l'identique.