# Métaheuristiques : Recherches locales et Algorithmes evolutionnaires

Master 1 informatique RIF IFI - Systèmes Artificiels Complexes

Sébastien Verel
verel@i3s.unice.fr
www.i3s.unice.fr/∼verel

Université Nice Sophia Antipolis
Laboratoire I3S
Equipe DOLPHIN - INRIA Lille Nord Europe

12 octobre 2012

# Plan

# Modélisation de Problèmes

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 3 | | | 7 | | | | |
| 6 | | | 1 | 9 | 5 | | | |
| | 9 | 8 | | | | | 6 | |
| 8 | | | | 6 | | | | 3 |
| 4 | | | 8 | | 3 | | | 1 |
| 7 | | | | 2 | | | | 6 |
| | 6 | | | | | 2 | 8 | |
| | | | 4 | 1 | 9 | | | 5 |
| | | | | 8 | | | 7 | 9 |

"petit" sudoko ($n = 3$)

Résolution d'un problème :

Problème $\Rightarrow$ modélisation $\Rightarrow$ solution(s)

Modélisation :

- simplification de la réalité (nombre de paramètres, "bruit", défauts...)

Conception d'un (bon) modèle :

- Connaissance experte du domaine
- Connaissance des méthodes de résolution (informatique)

# Problèmes d'optimisation combinatoire

- MAX - SAT
- Coloration de graphe
- Voyageur de commerce (TSP)
- Quadratic Assignment Problem (QAP)
- Scheduling
- paysages NK
- Sudoku (coloration de graphe)

# Problème SAT

Premier problème NP-difficile

- $N$ variables booléennes : $\{x_1, x_2, ..., x_N\}$,
- $m$ clauses : $\{C_1, C_2, ..., C_m\}$.
- $k_j$ littérals par clause $C_j$ : $\{l_{1,j}, l_{2,j}, ..., l_{k_j,j}\}$,

$$\bigwedge_{j=1}^{m} C_j$$

où $C_j = \bigvee_{i=1}^{k_j} l_{i,j}$ et $l_i = x_n$ ou $\bar{x_n}$

Existe-t-il une valeur des variables qui vérifie la formule logique ?

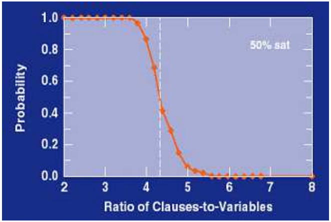
# SAT / MAX - SAT

Maximiser le nombre de clauses $C_j$ vérifiées

$$f(s) = \#\{C_j \ true\}$$

$s$ est une solution de SAT ssi $f(s) = m$

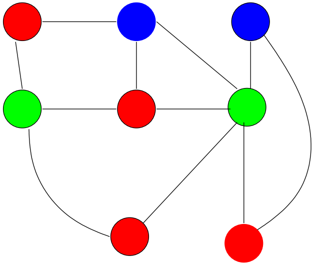
# SAT




Transition de phase suivant $\alpha = \frac{m}{N}$

Applications : vérification de circuits, logique, informatique...

# Coloration de graphe



Graphe $G = (S, A)$, $S$ ens. des sommets et $A$ ens. des arcs

Coloration $\alpha : S \rightarrow C$ telle que si $(s, t) \in A$ alors $\alpha(s) \neq \alpha(t)$

Applications : affection de fréquence en téléphonie mobile, emploi du temps, coloration des cartes...

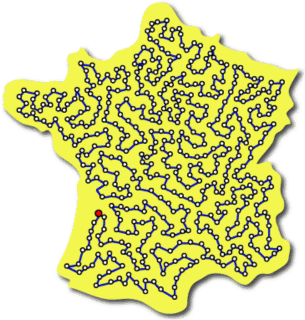
# Sodoku : coloration de graphe ?

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 3 | | | 7 | | | | |
| 6 | | | 1 | 9 | 5 | | | |
| | 9 | 8 | | | | | 6 | |
| 8 | | | | 6 | | | | 3 |
| 4 | | | 8 | | 3 | | | 1 |
| 7 | | | | 2 | | | | 6 |
| | 6 | | | | | 2 | 8 | |
| | | | 4 | 1 | 9 | | | 5 |
| | | | | 8 | | | 7 | 9 |

Graphe $G = (S, A)$,
$S = \{x_{i,j} \mid (i,j) \in \{1, \ldots, 3n\}^2\}$
$A = \{(x_{i,j}, x_{i,j'}) \ldots$
$(x_{i,j}, x_{i',j}) \ldots$
$(x_{i,j}, x_{i',j'}) \ \ldots\}$

Ensemble des couleurs $\{1, \ldots, n^2\}$
Remarque : "petit" problème puisque seulement 81 noeuds

# Voyageur de commerce (TSP)



Trouver le parcours le plus court passant par toutes les villes.

$$\sum_{i=1}^{n} \sum_{j=1}^{n} d_{x_i x_j}$$

avec :

- $n$ : nombre de villes
- $d_{rs}$ : distance entre les villes $r$ et $s$.
- $x$ une permutation :
  $x_i$ est la $i^{eme}$ ville traversée.

# Quadratic Assignment Problem (QAP)

- $n$ objets, $n$ emplacements
- $f_{ij}$ : flot entre objects $i$ et $j$,
- $d_{rs}$ : distance entre emplacement $r$ et $s$

Minimiser le flot total :

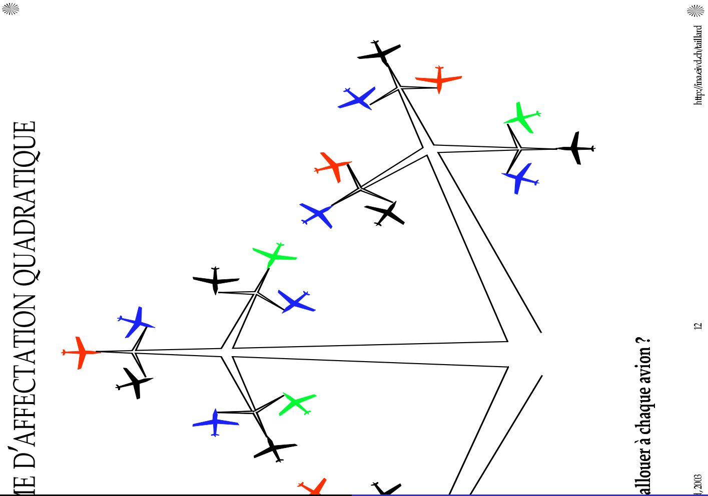
$$\sum_{i=1}^{n} \sum_{j=1}^{n} f_{ij} d_{p_i p_j}$$

avec $p$ une permutation :
$p_i$ emplacement l'objet $i$.

Applications : répartition de batiments ou de servives, affectation des portes d'aéroport, placement de modules logiques, claviers...

# Quadratic Assignment Problem (QAP)

Exemple d'aprés Taillard.



# Vehicule Routing Problem (VRP)

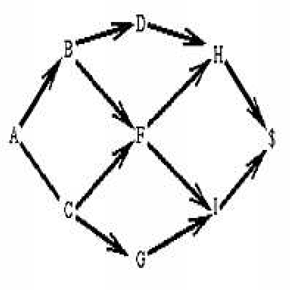
But : transporter des biens à des clients

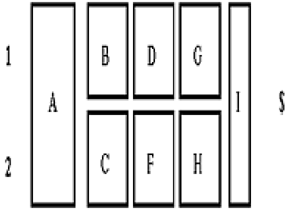
Véhicule à capacité limitée, fenêtre de temps, etc.

Problème : Déterminer pour chaque véhicule leur trajet.

Application : logistique du dernier kilomètre, etc.

# Job Scheduling Problem




Ensemble de taches
$J = \{j_1, j_2, j_3, \dots, j_p\}$
temps d'execution $p(j_i)$

Réalisés sur $m$ machines
$M = \{M_1, \dots, M_m\}$

Applications : ordonnancement, emploi du temps,...

# Paysages NK

$$f(x) = \frac{1}{N} \sum_{i=1}^{N} f_i(x_i; x_{i_1}, \ldots, x_{i_K})$$

- $N$ nombre d'acides animés
- $K \leq N - 1$ nombres d'intéraction entre ac. animés
- deux types d'ac. animé 0 ou 1
- $x_k$ le $k^{eme}$ ac. animé d'une chaine $x$
- $\{i_1, \ldots, i_K\} \subset \{1, \ldots, i-1, i+1, \ldots, N\}$
- $f_i : \{0,1\}^{K+1} \rightarrow [0,1]$

Application : modélisation de protéines

# exemple $N = 4 \ K = 2$

$$x = 0110$$

| $x_1 x_2 x_4$ | $f_1$ |
|---|---|
| 000 | 0.9 |
| 001 | 0.6 |
| 010 | 0.1 |
| 011 | 0.2 |
| ... | ... |

| $x_1 x_2 x_3$ | $f_2$ |
|---|---|
| 000 | 0.4 |
| 001 | 0.8 |
| 010 | 0.3 |
| 011 | 0.2 |
| ... | ... |

| $x_2 x_3 x_4$ | $f_3$ |
|---|---|
| 000 | 0.2 |
| ... | ... |
| 101 | 0.9 |
| 110 | 0.1 |
| 111 | 0.5 |

| $x_1 x_2 x_4$ | $f_4$ |
|---|---|
| 000 | 0.1 |
| 001 | 0.2 |
| 010 | 0.8 |
| 011 | 0.0 |
| ... | ... |

$$\begin{aligned} f(x) &= \frac{1}{4}( f_1(010) + f_2(011) + f_3(110) + f_4(010) ) \\ &= \frac{1}{4}( 0.1 + 0.2 + 0.1 + 0.8 ) \\ &= 0.3 \end{aligned}$$

# Optimization

## Inputs

- Search space : Set of all feasible solutions,
$$\mathcal{X}$$
- Objective function : Quality criterium
$$f : \mathcal{X} \rightarrow \mathbb{R}$$

## Goal

Find the best solution according to the criterium

$$x^{\star} = \text{argmax } f$$

# Optimization

## Inputs

- Search space : Set of all feasible solutions,

$$\mathcal{X}$$

- Objective function : Quality criterium

$$f : \mathcal{X} \rightarrow \mathbb{R}$$

## Goal

Find the best solution according to the criterium

$$x^{\star} = \text{argmax } f$$

*But, sometime, the set of all best solutions, good approximation of the best solution, good 'robust' solution...*

# Contexte

## Black box Scenario

We have only $\{(x_0, f(x_0)), (x_1, f(x_1)), ...\}$ given by an "oracle"
No information is either not available or needed on the definition of objective function

- Objective function given by a computation, or a simulation
- Objective function can be irregular, non differentiable, non continous, etc.

# Contexte

**Black box Scenario**

We have only $\{(x_0, f(x_0)), (x_1, f(x_1)), ...\}$ given by an "oracle"
No information is either not available or needed on the definition of objective function

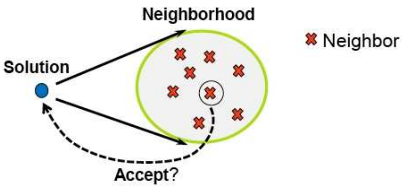
- Objective function given by a computation, or a simulation
- Objective function can be irregular, non differentiable, non continous, etc.
- (Very) large search space for discrete case (combinatorial optimization), *i.e.* NP-complete problems

# Search algorithms

## Principle

**Enumeration of the search space**

- A lot of ways to enumerate the search space
- Using random sampling : Monte Carlo technics
- Local search technics :

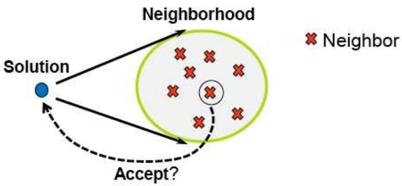


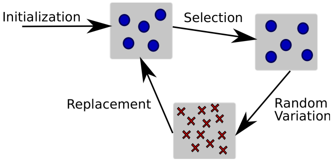
# Search algorithms

## Principle

**Enumeration of the search space**

- A lot of ways to enumerate the search space
- Using random sampling : Monte Carlo technics
- Local search technics :



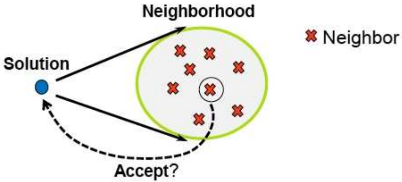



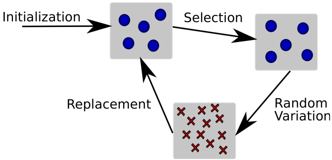
# Search algorithms

## Principle

**Enumeration of the search space**

- A lot of ways to enumerate the search space
- Using random sampling : Monte Carlo technics
- Local search technics :




- If objective function $f$ has no propertie : random search
- If not...

# Retour à MAX-SAT

Comment résoudre ce genre de problèmes ?

$$(x_1 \vee x_2) \wedge (\bar{x_2} \vee x_3) \wedge (x_1 \vee \bar{x_3})$$

$$(x_4 \vee x_2 \vee x_3) \wedge (x_2 \vee x_4 \vee x_5) \wedge (\bar{x_2} \vee x_1 \vee x_5) \wedge (\bar{x_2} \vee x_5 \vee x_4) \wedge (x_1 \vee \bar{x_3} \vee \bar{x_4})$$

# Retour à MAX-SAT

mathématiquement ou ...

- Exhaustivement $n = 20$, $n = 100$, ...
- Aléatoirement
- Construction de solution (pas dans ce cours)
- Méthodes exactes (pas dans ce cours)

ou ...

# Heuristiques

Heuristique : algorithme de résolution basé sur l"'expérience" ne fournissant pas nécessairement une solution optimale

On désire toutefois :

- Le plus souvent possible une solution proche de l'optimalité
- Le moins souvent possible un mauvaise solution (différent !)
- Une complexité "raisonnable"
- De la simplicité d'implémentation (code light en version de base...)

# Metaheuristiques

Métaheuristique : ensemble d'heuristiques
Peu probable qu'un algorithme puisse résoudre tout problème

Métaheuristique :

- regroupe des heuristiques dépendant de paramètres
- décrit une méthode de conception d'heuristique

*de*
*un aveu d'impuissance*
*à*
*des techniques performantes d'optimisation difficile*

# Metaheuristisques de recherche locale

Algorithmes à population de solutions

- Algorithmes Evolutionnaires (EA) : Holland 1975 et même avant
- Algorithmes d'essaims particulaires (PSO) : R. Ebenhart et J. Kennedy 1995.
- Algorithmes de fourmis (ACO) : Bonabeau 1999

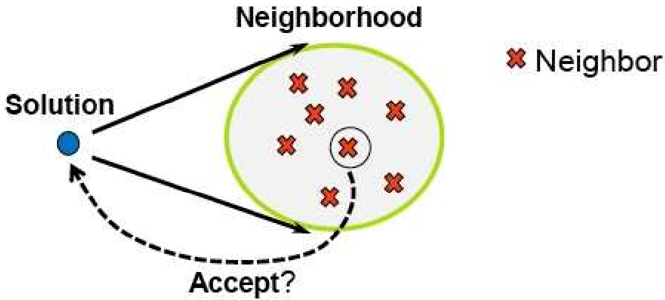
# Metaheuristisques de recherche locale

Algorithmes à solution unique

- (Recherches aléatoire),
- Algorithmes de descente : Hill-Climber (HC), première-descente
- Recuit Simulé (SA) : Kirkpatrick *et al* 1983,
- Recherche Tabou (TS) : Glover 1986 - 89 -90,

# Stochatic algorithms with unique solution (Local Search)

- $\mathcal{S}$ set of solutions (search space)
- $f : \mathcal{S} \rightarrow \mathbb{R}$ objective function
- $\mathcal{V}(s)$ set of neighbor's solutions of $s$



# Recherche Locale (LS)

3 ingrédients indispensables :

- $\mathcal{S}$ ensemble des solutions (espace de recherche),
- $f : S \rightarrow \mathbb{R}$ fonction objectif à maximiser (ou coût à minimiser)
- $\mathcal{V}(s)$ ensemble des solutions voisines de $s$

*Algorithme d'une Recherche Locale*

Choisir solution initiale $s \in \mathcal{S}$
**répéter**
  choisir $s^{'} \in \mathcal{V}(s)$
  $s \leftarrow s^{'}$
**jusqu'à** critère d'arrêt non verifié

# Un exemple très simple

Maximiser $f(x) = \sum_{i=1}^{N} x_i$ où $x$ chaine binaire de taille $N$.
Une solution pour $N = 6$ :

$$x = 01101 \text{ et } f(x) = 3$$

- Taille de l'espace de recherche $\mathcal{S}$ :

# Un exemple très simple

Maximiser $f(x) = \sum_{i=1}^{N} x_i$ où $x$ chaine binaire de taille $N$.
Une solution pour $N = 6$ :

$$x = 01101 \text{ et } f(x) = 3$$

- Taille de l'espace de recherche $\mathcal{S}$ : $2^N$

# Un exemple très simple

Maximiser $f(x) = \sum_{i=1}^{N} x_i$ où $x$ chaine binaire de taille $N$.
Une solution pour $N = 6$ :

$$x = 01101 \text{ et } f(x) = 3$$

- Taille de l'espace de recherche $\mathcal{S}$ : $2^N$

## Exercice

- Coder, dans le langage que vous voulez, la fonction d'évaluation.
- Tester en affichant une solution et la valeur de la fonction
- Coder la recherche aléatoire qui consiste à générer des solutions aléatoirement (uniformément)
- Evaluer les performances de la recherche aléatoire

# Recherche Locale Aléatoire

Heuristique d'exploration maximale

*Recherche locale aléatoire*

Choisir solution initiale $s \in \mathcal{S}$
**répéter**
  choisir $s^{'} \in \mathcal{V}(s)$ aléatoirement
  $s \leftarrow s^{'}$
**jusqu'à** Nbr d'éval. $\leq$ maxNbEval

- Algorithme inutilisable en pratique
- Algorithme de comparaison
- Opérateur local de base de nombreuses métaheuristiques

# Un exemple très simple

- Voisinage de $x \in \{0, 1\}^N$ :

# Un exemple très simple

- Voisinage de $x \in \{0,1\}^N$ :
  $\mathcal{V}(x)$ : ensemble des chaînes binaires à une distance (de Hamming) 1.

Pour $x = 01101$, $\mathcal{V}(x) = \{$

# Un exemple très simple

- Voisinage de $x \in \{0,1\}^N$ :
  $\mathcal{V}(x)$ : ensemble des chaînes binaires à une distance (de Hamming) 1.

Pour $x = 01101$, $\mathcal{V}(x) = \{$

- 01100
- 01111
- 01001
- 00101
- 00101
- 11101 $\}$

# Un exemple très simple

- Voisinage de $x \in \{0,1\}^N$ :
  $\mathcal{V}(x)$ : ensemble des chaînes binaires à une distance (de Hamming) 1.

Pour $x = 01101$, $\mathcal{V}(x) = \{$

- 01100
- 01111
- 01001
- 00101
- 00101
- 11101 $\}$

- Taille du voisinage de chaque solution $x$ :

# Un exemple très simple

- Voisinage de $x \in \{0,1\}^N$ :
  $\mathcal{V}(x)$ : ensemble des chaînes binaires à une distance (de Hamming) 1.

Pour $x = 01101$, $\mathcal{V}(x) = \{$

- 01100
- 01111
- 01001
- 00101
- 00101
- 11101 $\}$

- Taille du voisinage de chaque solution $x$ : $N$

# Un exemple très simple

## Exercice

- Coder la recherche locale aléatoire
- Afficher le graphique du parcours de la recherche (nb d'évaluations *vs.* valeur fonction)

# Hill-Climbing (HC) (ou steepest-descent)

Heuristique d'exploitation maximale.

*Hill Climbing*

Choisir solution initiale $s \in \mathcal{S}$
**répéter**
  Choisir $s^{'} \in \mathcal{V}(s)$ telle que $f(s^{'})$ est maximale
  $s \leftarrow s^{'}$
**jusqu'à** $s$ optimum local

- Algorithme de comparaison
- Opérateur local de base de métaheuristique

# Une variante : first-improvement

*Première amélioration*

Choisir solution initiale $s \in \mathcal{S}$
**répéter**
  Choisir $s^{'} \in \mathcal{V}(s)$ aléatoirement
  **si** $f(s) \leq f(s^{'})$ **alors**
    $s \leftarrow s^{'}$
  **fin si**
**jusqu'à** $s$ optimum local OU nbr d'éval. $\leq$ maxNbEval

# Un exemple très simple

## Exercice

- Coder la recherche first-improvement
- Evaluer les performances de la recherche first-improvement

## Hill-Climbing (HC) (ou steepest-descent)

Quel est l'inconvénient majeur du Hill-Climbing ?

# Hill-Climbing (HC) (ou steepest-descent)

Peut-on imaginer des situations où ce n'est qu'un faible inconvénient ?

# Un exemple très simple

## Exercice

- Changer la fonction d'évaluation (UBQP) :

$$f(x) = \sum_{i=1}^{N} \sum_{j=1}^{N} q_{ij} x_i x_j$$

avec $q_{ij} = (-1)^{3i+j} \times 2^{(7*i+j) mod 16}$

- Evaluer les performances de la recherche first-improvment.

# Points critiques dans la conception d'une recherche locale

- Représentation des solutions :
  - codage plus ou moins redondant,
  - introduction dans le codage de connaissances au problème,
  - complexité du codage, de l'évaluation (évaluation incrémentale),
  - etc.

# Points critiques dans la conception d'une recherche locale

- représentation des solutions : redondance, introduction connaissance du problème, complexité...
- voisinage : taille, continuité :
  $s^{'} \in \mathcal{V}(s), \delta(f(s^{'}), f(s)) \leq \epsilon$
  $\mathcal{V}(s) = \{s^{'} \mid s^{'} = op(s)\}$
- fonction $f$ guide vers l'optimalité
- "choisir" un voisin

Autres : générateur aléatoire ( !), choix d'implémentation (évaluation incrémentale), critère d'arrêt...

# Points critiques dans la conception d'une recherche locale

- Voisinage :
  - taille :
    $\mathcal{V}(s) = \{s^{'} \mid s^{'} = op(s)\}$
  - continuité :
    Pour tout $s \in \mathcal{S}$ et $s^{'} \in \mathcal{V}(s)$,
    $P(\delta(f(s^{'}), f(s)) \leq \epsilon)$ est grande
  - une idée pour bon voisinage :
    prob. trouver meilleure solution dans le voisinage $>$ prob. trouver meilleure solution à l'extérieur

# Points critiques dans la conception d'une recherche locale

- fonction $f$ guide vers l'optimalité :
  plus $f(x)$ est grand, plus $x$ est proche de l'optimum.
- "choisir" un voisin :
  Choisir un voisin tel que la probabilité de trouver une bonne solution depuis ce voisin est grande.

Autres : générateur aléatoire ( !), choix d'implémentation (évaluation incrémentale), critère d'arrêt...

# Metaheuristics

## Random search / Hill Climbing

**Algorithme 1** Random walk

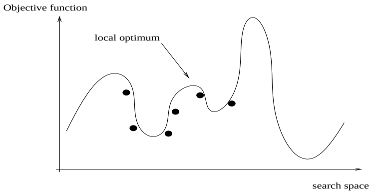
Choose randomly initial solution $s \in \mathcal{S}$
**répéter**
  Choose $s^{'} \in \mathcal{V}(s)$ randomly
  $s \leftarrow s^{'}$
**jusqu'à** ...

**Algorithme 2** Hill-climbing

Choose randomly initial solution $s \in \mathcal{S}$
**répéter**
  Choose $s^{'} \in \mathcal{V}(s)$ such as $f(s^{'})$ is maximal
  $s \leftarrow s^{'}$
**jusqu'à** $s$ local optimum

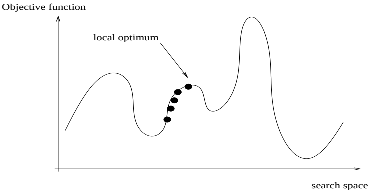
# Metaheuristics
## Random search / Hill Climbing

Random walk




maximal exploration , diversification

Hill-climbing




maximal exploitation , intensification

**Main issue : exploration / exploitation tradeoff**

escape from local optima, etc.

$\Longrightarrow$ simulated annealing, tabu search

# Recuit Simulé (Simulated Annealing)

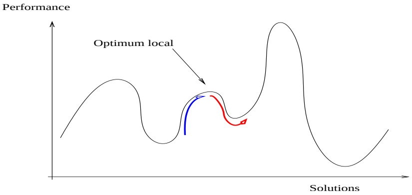
Utilisé depuis les années 80,

- Metropolis (1953) simulation du refroidissement de matériaux (Thermodynamique)
- Kirkpatrick *et al* (IBM 1983) utilisation pour la résolution de problème d'optimisation.

But : échapper aux optima locaux
Principe : probabilité non nulle de sélection d'une solution voisine dégradée

# Recuit Simulé : analogie

| Système physique | Problème d'optimisation |
|---|---|
| Energie | fonction objectif |
| Etats du système | solution |
| États de basse énergie | bonne solution |
| Température | paramètre de contrôle |



# Recuit Simulé

Choisir solution initiale $s \in \mathcal{S}$ et temperature initiale $T$
**répéter**
  choisir aléatoirement $s^{'} \in \mathcal{V}(s)$, $\Delta = f(s^{'}) - f(s)$
  **si** $\Delta > 0$ **alors**
    $s \leftarrow s^{'}$
  **sinon**
    $u$ nombre aléatoire de $[0, 1]$
    **si** $u < e^{\frac{\Delta}{T}}$ **alors**
      $s \leftarrow s^{'}$
    **fin si**
  **fin si**
  update temperature $T$
**jusqu'à** Critère d'arrêt vérifié

# Recuit Simulé : remarques

Si $\Delta < 0$ alors la probabilité $\exp(\frac{\Delta}{T})$ est proche de 0 lorsque :

- la différence $|\Delta = f(s^{'}) - f(s)|$ est grande
- la temperature est petite

Conséquences :

- lorsque température grande (début de la recherche) :
  $\longrightarrow$ recherche aléatoire
- lorsque température petite (fin de la recherche) :
  $\longrightarrow$ Hill-Climbing

# Recuit Simulé : température initiale

Evaluer $\Delta_0 = f(s_0^{'}) - f(s_0)$ :

- Choisir $n$ (grand si possible) solutions aléatoires initiales $s_0$ et une solution voisine $s_0^{'}$
- calculer la moyenne de $\Delta_0$ sur l'échantillon

Température initiale $T_0$ telle que $\tau_0 = e^{\frac{\Delta_0}{T_0}}$ désiré :
qualité "médiocre" ($\tau_0 = 0.50$) : démarrage à haute température
qualité "bonne" ($\tau_0 = 0.20$) : démarrage à basse température

# Recuit Simulé : décroissance de "température"

décroissance suivant une loi géométrique $T_{k+1} = \alpha T_k$
souvent $0.8 \leq \alpha < 1.0$

Changement par pallier de température suivant l'une des deux conditions :

- $12.N$ perturbations acceptées (mouvements de solution)
- $100.N$ perturbations tentées (mouvement ou non mouvement)

où $N$ est un paramètre qui décrit la taille du problème (nombre de villes, de variables...)

# Recuit Simulé : Critère d'arrêt

Arrêt aprés 3 palliers successifs sans aucune acceptation.

## Recuit Simulé : Remarques

- Toutes ces indications ne sont pas universelles : L'analyse du problème et l'expérience de concepteur permettent de les adapter
- Vérifier votre générateur aléatoire
- La qualité du résultat doit dépendre "peu" de l'exécution de l'algorithme

Premières Applications : dans le placement de circuits électroniques

# Recuit Simulé : Bibliographie

E. Aarts, J. Korst : « Simulated Annealing and Boltzmann machine » John Wiley, New-York 1989

P. Siarry : « La méthode du recuit simulé : théorie et application » ESPCI - IDSET , 10 rue Vauquelin, Paris 1989

# Recherche Tabou (Tabu Search)

Introduite par Glover en 1986 :
«Future paths for Integer Programming and Links to Artificial Intelligence», Computers and Operations Research, 5 :533-549, 1986.

But : échapper aux optima locaux
Principe : Introduction d'une notion de mémoire dans la stratégie d'exploration

*Interdiction de reprendre des solutions déjà (ou récemment) rencontrées*

# Recherche Tabou (Tabu Search)

Choisir solution initiale $s \in \mathcal{S}$
Initialiser Tabou $M$
**répéter**
  choisir $s^{'} \in \mathcal{V}(s)$ telle que :
  ( $f(s^{'})$ meilleure solution de $\mathcal{V}(s)$ ET Critère d'aspiration vérifié )
  OU $f(s^{'})$ meilleure solution de $\mathcal{V}(s)$ non taboue
  $s \leftarrow s^{'}$
  update Tabou $M$
**jusqu'à** Critère d'arrêt vérifié

# Recherche Tabou : mémoire des tabous

Les tabous sont souvent des mouvements tabous pendant une durée

*exemple* : problème maxsat avec $n = 6$
$M = (0, 3, 0, 0, 0, 0)$
le deuxième bit ne peut être modifié pendant 3 itérations.
$M = (1, 2, 0, 0, 2, 5)$
seuls bits non tabou 3 et 4

Lorsqu'un mouvement est effectué :
interdiction pendant $n$ itérations

# Exercice Tabou

Exécuter un Tabou sur un problème MAX-SAT.

# Recherche Tabou : mémoire des tabous

Lorsqu'un mouvement est effectué :
interdiction pendant $n$ itérations
Si $n$ trop faible, tabou peu efficace
Si $n$ trop grand, les solutions sont "à flanc de coteau".

$\longrightarrow$ Stratégie de diversification

# Recherche Tabou : Mémoire à long terme

Statistique sur les mouvements :
Repérer les mouvements trop utilisés (difficulté de recherche, optimum local...)

Fréquence $freq(m)$ d'utilisation d'un mouvement $m$ :
pénalisation du mouvement $m$ par ajout d'interdiction en fonction de $freq(m)$.

# Recherche Tabou : Critère d'aspiration

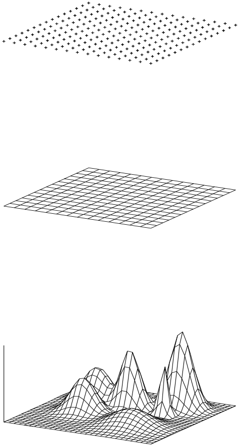
Enlever le caractère tabou d'une solution :

Lorsque la solution est la meilleure jamais rencontrée

# Recherche Tabou : Bibliographie

Glover *et al* : «Tabu Search» Kluwer Academic Publishers, 1997

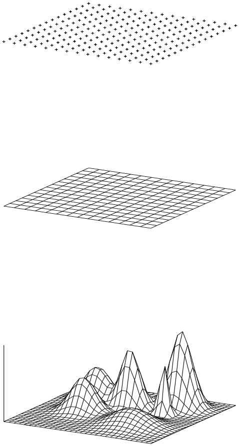
# Paysage Adaptatif



Origine Biologique (Wright 1930) : Modélisation évolution des espèces

Utiliser pour modéliser des systèmes dynamiques : physique statistique, évolution moléculaire, écologie, etc

# Optimisation combinatoire



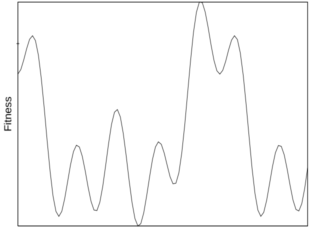
*Paysage adaptatif* $(\mathcal{S}, \mathcal{V}, f)$ :

- $\mathcal{S}$ : ensemble de solutions potentielles,
- $\mathcal{V} : \mathcal{S} \rightarrow 2^{\mathcal{S}}$ : relation de voisinage,
  - $\mathcal{V}(x) = \{y \mid y = op(x)\}$
  - $\mathcal{V}(x) = \{y \mid d(y, x) \leq 1\}$
- $f : \mathcal{S} \rightarrow \mathbb{R}$ : fonction à optimiser.

# Intérêts du concept

- Relation entre description géométrique d'un problème et dynamique de recherche
- Pertinence du choix de l'opérateur
- Connaissance de la géométrie du problème
  $\Rightarrow$ conception de métaheuristiques adaptées

# Paysage Multimodal




paysage multimodal

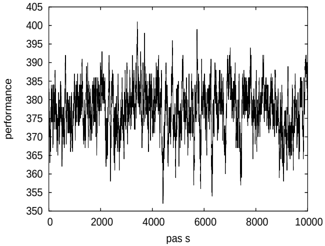
Optimum local : aucune solution voisine de meilleure performance.

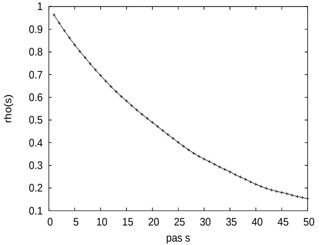
- Difficulté liée au nombre
- Taille des bassins d'attraction

Estimation : Marche adaptative $(s_0, s_1, \ldots)$ où $s_{i+1} \in \mathcal{V}(s_i)$, $f(s_i) < f(s_{i+1})$

- Terminaison sur optimum local
- Longueur : indice de distance inter-optima

# Paysage Rugueux

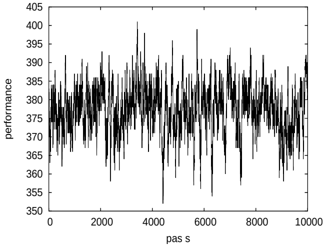


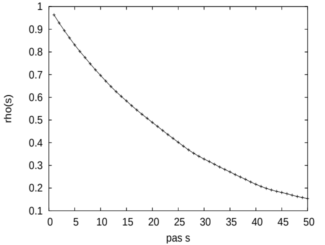




Autocorrélation lors d'une marche aléatoire (Weinberger 1996)
Longueur de corrélation $\tau = \frac{1}{\rho(1)}$

- $\tau$ petit : paysage rugueux
- $\tau$ grand : paysage lisse

# Paysage Rugueux




Autocorrélation lors d'une marche aléatoire (Weinberger 1996)
Longueur de corrélation $\tau = \frac{1}{\rho(1)}$

- $\tau$ petit : paysage rugueux
- $\tau$ grand : paysage lisse

conjecture
(Stadler 92, Garcia 97) :

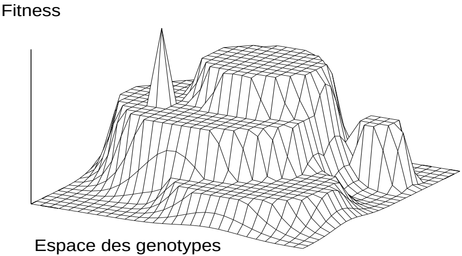
$$M \approx |\mathcal{S}|/|\mathcal{B}(x, \tau)|$$

# Paysage Neutre

Théorie de la neutralité (Kimura $\approx$ 1960)

Théorie de la mutation et de la dérive aléatoire

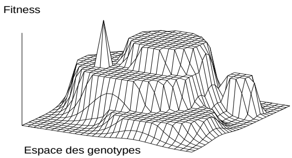
Rôle prépondérant des mutations sans influence sur la performance




- Géométrie de plateaux
- Degré de neutralité
- Réseaux de neutralité (Schuster 1994, structure secondaire de l'ARN)

# Paysage Neutre
## Évolution artificielle

Prise en compte depuis les années 80 en évolution artificielle : redondance (Goldberg 87)




Présence dans :

- Programmation génétique
- Contrôleur de robot
- Conception de circuit (Cartesian GP)
- Étiquetage de graphe (MinLA)

# Paysage neutre
## Optimisation combinatoire

Plusieurs possibilités :

- Diminuer la neutralité :
  conjecture : redondance nuit aux performances
- Utiliser une métaheuristique adaptée :
  conjecture : neutralité est intrisèque
- Augmenter la neutralité par un choix de codage redondant :
  conjecture : éviter les optima locaux

# Paysage neutre

## Optimisation combinatoire

Plusieurs possibilités :

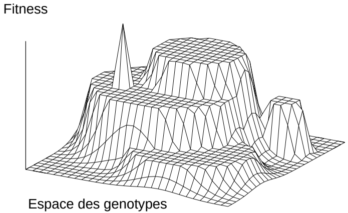
- Diminuer la neutralité :
  conjecture : redondance nuit aux performances
- Utiliser une métaheuristique adaptée :
  conjecture : neutralité est intrisèque
- Augmenter la neutralité par un choix de codage redondant :
  conjecture : éviter les optima locaux

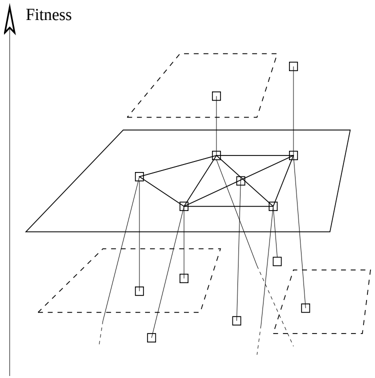
**Meilleure description et connaissance des paysages neutres**

$\longrightarrow$ Concevoir de nouvelles métaheuristiques

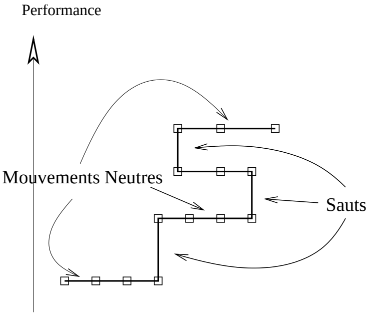
$\longrightarrow$ Évaluer la pertinence d'un codage

# Réseaux de neutralité (Schuster 1994)

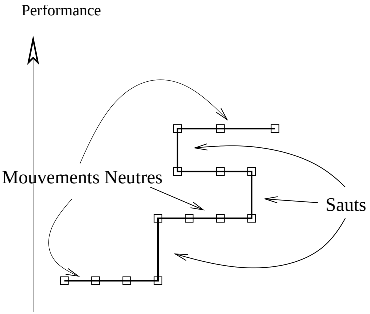


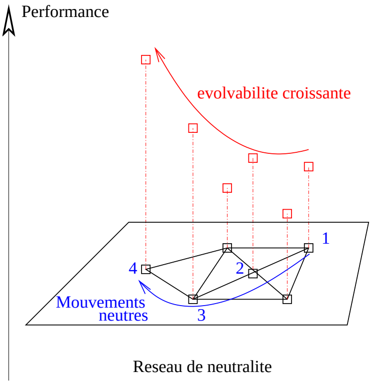



# Recherche Périscopique (Scuba Search)



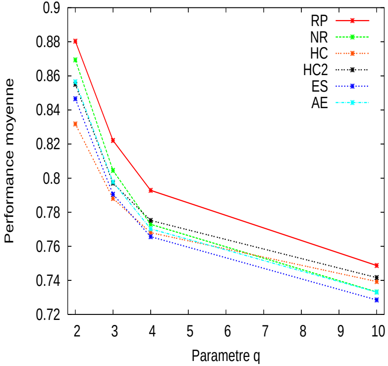
# Recherche Périscopique (Scuba Search)



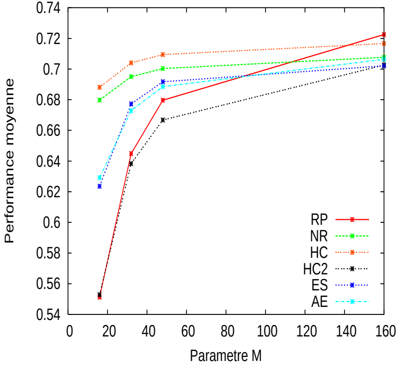




Mouvements neutres guidés par l'évolvabilité

# Performances moyennes sur $N = 64$ et $K = 4$




paysage $NK_q$




paysage $NK_M$

# Conclusion

- Problèmes d'optimisation combinatoire fréquents dans l'industrie (et fondamentaux en informatique théorique)
- Métaheuristiques de recherche locale :
  $\longrightarrow$ Recherche aléatoire
  $\longrightarrow$ Hill-climbing et first-ascent
  $\longrightarrow$ Recuit simulé
  $\longrightarrow$ Recherche tabou
- Paysage de Fitness : métaphore qui permet
  $\longrightarrow$ l'analyse du lien entre métaheurisque et problème
  $\longrightarrow$ imaginer de nouvelles metaheurisques (recherche périscopique)
- Propriété principale :
  $\longrightarrow$ rugosité : optima locaux, structure de corrélation
  $\longrightarrow$ neutralité : réseau de neutralité