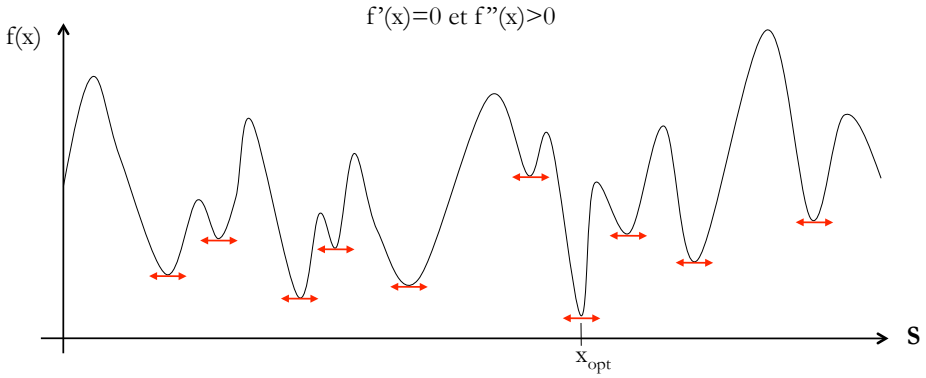
MÉTAHEURISTIQUES

- **Trouver le minimum d'une fonction**

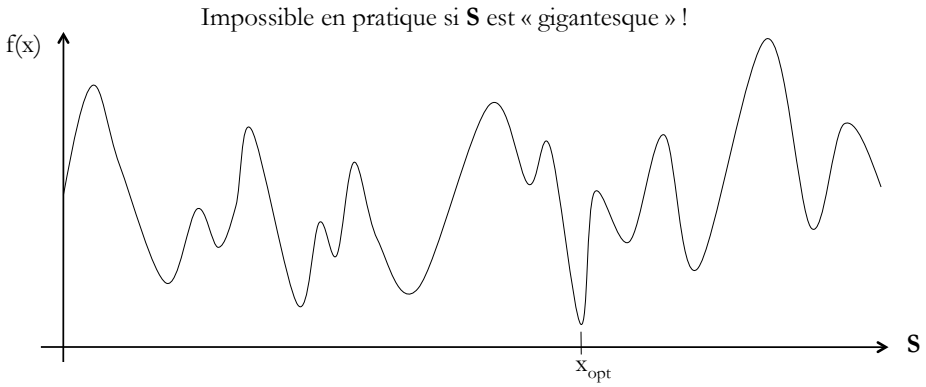
  Cas hors d'étude d'une fonction analytique dérivable (**S** non fini)



MÉTAHEURISTIQUES

- **Trouver le minimum d'une fonction**

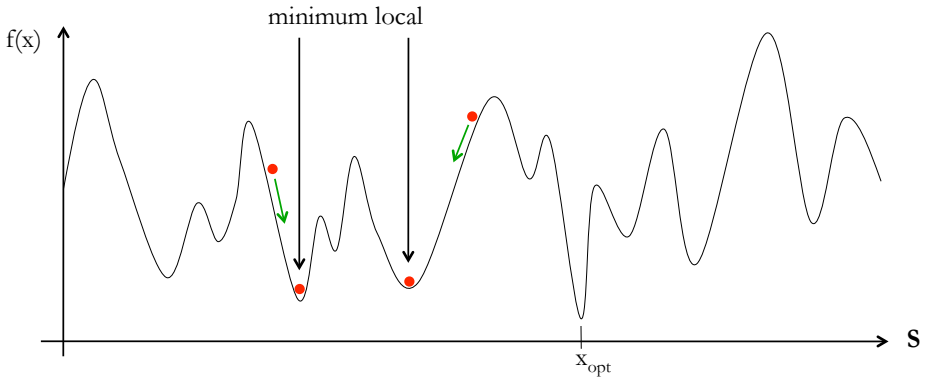
  Explorer toutes les solutions de **S** ?



MÉTAHEURISTIQUES

- **Trouver le minimum d'une fonction**

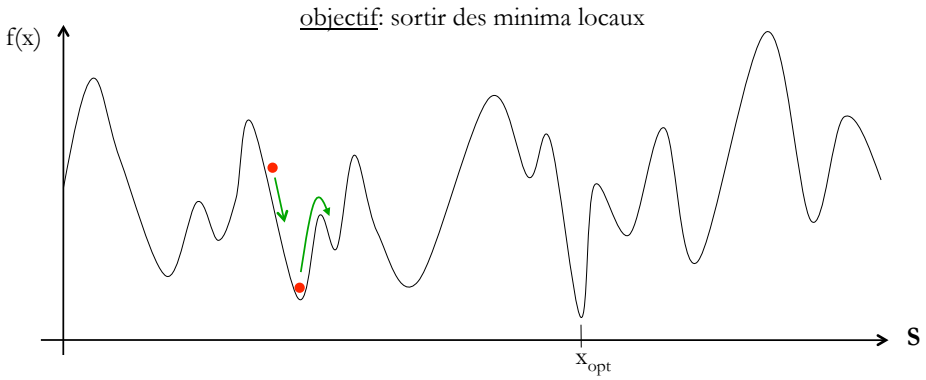
  Les **méthodes de descente**



MÉTAHEURISTIQUES

- **Trouver le minimum d'une fonction**

  Le **recuit simulé** ou la **méthode Tabou**

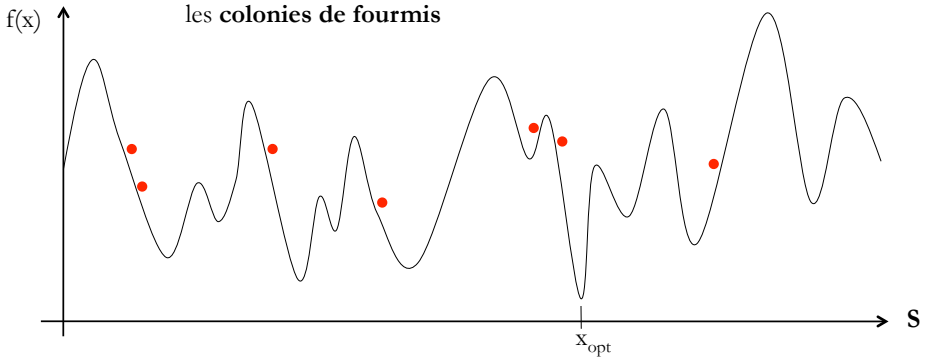


MÉTAHEURISTIQUES

- **Trouver le minimum d'une fonction**

  Les méthodes à base de population

les **algorithmes évolutionnaires** (génétiques, …)
les **colonies de fourmis**



MÉTAHEURISTIQUES


Computers & Operations Research 39 (2012) 1074–1086

Contents lists available at ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/caor

ELSEVIER

Metaheuristics for the traveling salesman problem with pickups, deliveries and handling costs

Günes Erdoğan [a,*], Maria Battarra [b], Gilbert Laporte [c], Daniele Vigo [d]


Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

ELSEVIER

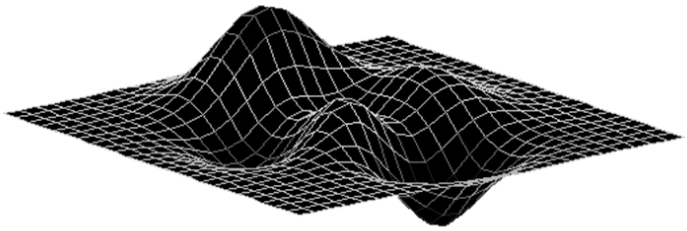
Location discovery in Wireless Sensor Networks using metaheuristics

Guillermo Molina*, Enrique Alba

LANDSCAPE

- **Notion de Landscape**

  Un **landscape** est la donnée d'un ensemble **S**, d'une fonction de fitness f(.) et d'une relation de voisinage V.



VOISINAGE

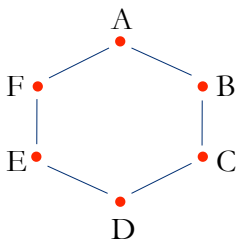
- **Notion de Voisinage**

  Soit x∈**S**, Qu'est-ce qu'un voisin de x ?

- **Exemple : le « voyageur de commerce »**

  6 villes notées A, B, C, D, E et F.

  Soit x un trajet de **S**, x = « ABCDEF » (*notation circulaire*)



VOISINAGE

- **Transformation locale**

  Transformation « échanger 2 villes » :

  x = « ABCDEF » ⟶ x' = « ADCBEF »

  V(x) = { BACDEF, CBADEF, DBCAEF, EBCDAF, FBCDEA,
  ACBDEF, ~~ADCBEF~~, AECDBF, AFCDEB, ABDCEF,
  ~~ABEDCF~~, ABFDEC, ABCEDF, ~~ABCFED~~, ABCDFE }

VOISINAGE

- **Transformation locale**

  Transformation « insértion-décalage » :

  x = « ABCDEF » ⟶ x' = « AEBCDF »

  x = « ABCDEF » ⟶ x' = « ACDEBF »

  Cette transformation définie un autre ensemble de voisins
  ⇒ un autre landscape.

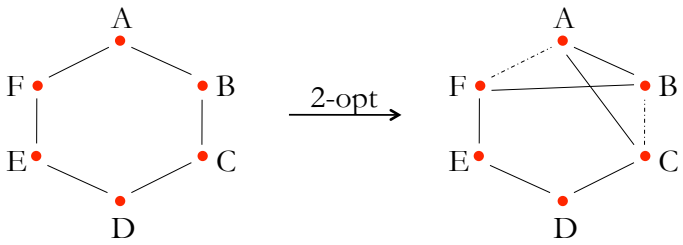
  Transformation « inversion » :

  x = « ABCDEF » ⟶ x' = « AEDCBF »

VOISINAGE

- **Transformation locale**

Transformation 2-opt : « échange de 2 arêtes disjointes »

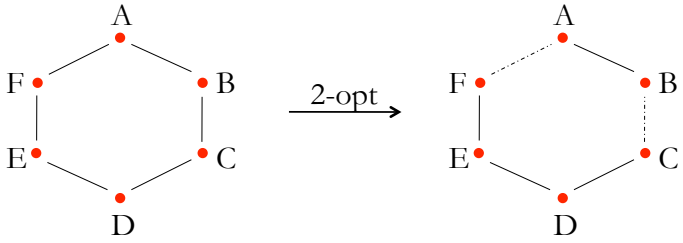


x = « ABCDEF »

x' = « ABFEDC »

VOISINAGE

- **Transformation locale**



V(x)= {ABFEDC, ABCFED, ABCDFE, ACBDEF, ADCBEF, AEDCBF, ABDCEF, ABEDCF, ABCEDF }

pour n villes, $|V(x)| = \frac{n(n-3)}{2}$

Il existe une transformation 3-opt : « échange de 3 arêtes disjointes »

VOISINAGE

- **Exemple : le « sac à dos »**

  Soit **S** l'ensemble des solutions possibles (ensemble des sous-ensembles d'items).

  Soit x une solution de **S**,

  x = « 011001011101001101001010 »

- **Fitness**

  f(x) = somme des utilités des objets à 1

- **Transformation locale**

  Inverser un bit x = « 01100101 »

  V(x) = { 11100101, 00100101, 01000101, 01110101, 01101101,
  01100001, 01100111, 01100100 }

VOISINAGE

- **Exemple : coloration d'un graphe**

  Soit x une solution du problème :

  x = [c(1), …, c(i), …, c(n)]

  où n est le nombre de sommets et c(i) est la couleur du sommet i.
  Pour avoir une seule représentation d'une solution x, les couleurs sont systématiquement ordonnées par ordre croissant.

- **Fitness**

  f(x) = nombre d'arêtes avec des sommets de même couleur (nombre d'erreurs)

- **Transformation locale**

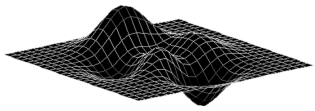
  Changer la couleur d'un sommet. x = [1221]

  pour n sommets et k couleurs, $|V(x)| = n(k-1)$

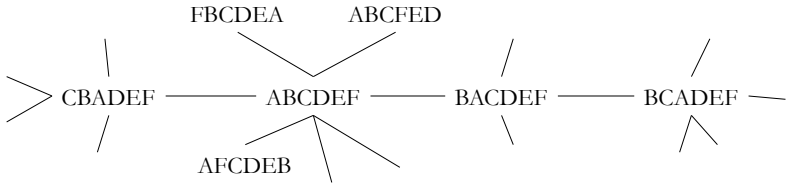
LANDSCAPE

- **Représentation graphique du landscape**

Généralement impossible de représenter le landscape sous cette forme :



Le landscape est un graphe (orienté en général), dont les sommets sont valués par la fitness :



LANDSCAPE

- **Conclusion de la thèse de Lionel Barnett (2003)**

« the more we know of the **statistical properties** of a class of fitness landscapes, the better equipped we will be for the **design of effective search algorithms** for such landscapes »

- **Principales propriétés étudiées**

  - Nombre et Taille des bassins d'attraction
  - Rugosité ou aspect lisse (étude de l'autocorrélation)
  - Degré de neutralité : nombre de solutions voisines avec la même fitness

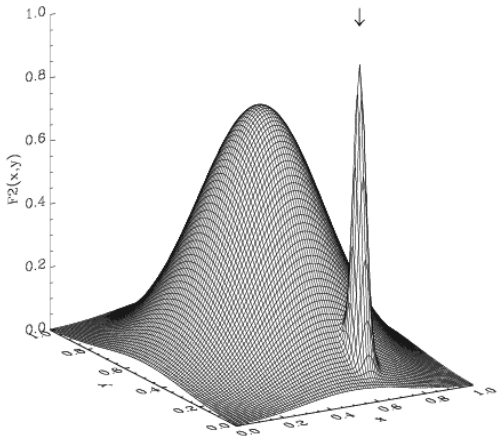
A Search Space "Cartography" for Guiding Graph Coloring Heuristics

Daniel Cosmin Porumbel [1,2], Jin Kao Hao [1] and Pascale Kuntz [2]
1) LERIA, Université d'Angers, 2 bd. Lavoisier, 49045 Angers, France
2) LINA, Polytech'Nantes, rue Christian Pauc, 44306 Nantes, France

August 7, 2009

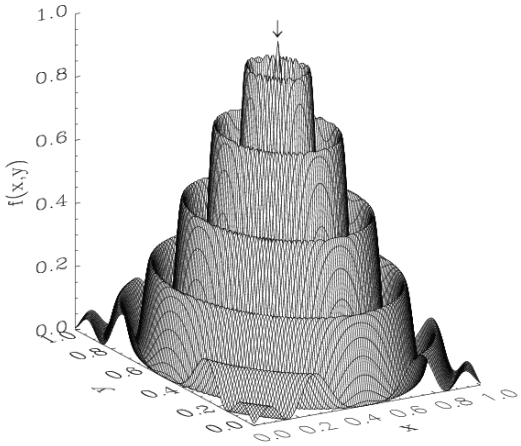
LANDSCAPE

- **Exemple de Landscape**



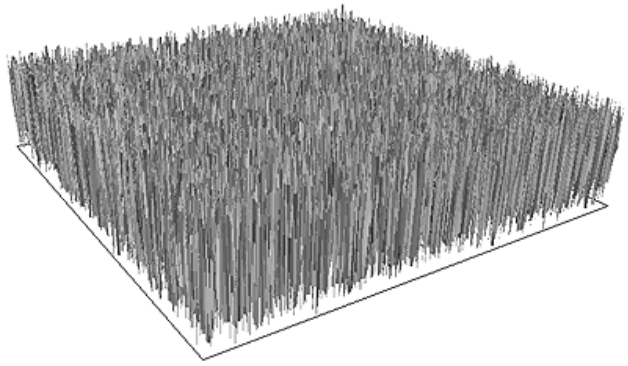
LANDSCAPE

- **Exemple de Landscape**



LANDSCAPE

- **Exemple de Landscape**

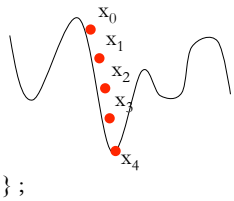


MÉTHODE DE DESCENTE (HILL-CLIMBING)

- La **méthode de descente** est une **méthode gloutonne**

construction d'une suite de solution x_0, x_1, x_2, …, jusqu'à un minimum locale.

toutes les solutions voisines sont moins bonnes



x_0 : solution initiale ;
i=-1 ;
répéter
{ i=i+1 ;
Choisir $x_{i+1} \in V(x_i)$ tel que $f(x_{i+1})=\min\{ f(y) / y \in V(x_i) \}$;
}
jusqu'à $f(x_{i+1})>f(x_i)$;

dépend de x_0 ⇒ à exécuter plusieurs fois avec des x_0 différents.

MÉTHODE DE DESCENTE (HILL-CLIMBING)

- **Bassin d'attraction**

  Le **bassin d'attraction** d'une solution x* est l'ensemble des solutions x tel que l'algorithme Hill-Climbing mène à x* :

  $$\{ x \in S \ / \ \text{Hill-Climbing}(x) = x^* \}$$

- **Etude du landscape**

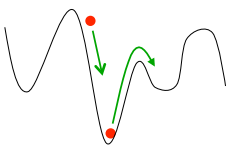
  Plus la taille du bassin d'attraction de l'optimum global est petite, plus le temps nécessaire pour le trouver est long !

MÉTHODE DU RECUIT SIMULÉ (SIMULATED ANNEALING)

- Le **recuit simulé** est une méthode issue de la thermodynamique (fabrication de cristaux)

  Objectif : sortir des minima locaux en acceptant, sous certaines conditions, des solutions « moins bonnes ».

  Besoin d'une **température**



  Principe :

  - à chaque x_i , choisir x_{i+1} dans $V(x_i)$
  - si $f(x_i) \geq f(x_{i+1})$, accepter x_{i+1}.
  - si $f(x_i) < f(x_{i+1})$, accepter x_{i+1} si l'écart entre $f(x_i)$ et $f(x_{i+1})$ n'est pas trop grand et si la température n'est pas trop faible.

MÉTHODE DU RECUIT SIMULÉ (SIMULATED ANNEALING)

x_0 : solution initiale ; t_0 : température initiale ;
$x_{min} = x_0$; $f_{min} = f(x_{min})$; $i = 0$;

pour k=0 **à** n_1 **faire**
{ **pour** l=1 **à** n_2 **faire**
{ Choisir $y \in V(x_i)$;
$\Delta f = f(y) - f(x_i)$;
si ($\Delta f \leq 0$) **alors** { $x_{i+1} = y$;
si ($f(x_{i+1}) < f_{min}$) **alors** { $f_{min} = f(x_{i+1})$; $x_{min} = x_{i+1}$; }
}
sinon { tirer $p \in [0,1]$ avec une distribution uniforme ;
si ($p \leq \exp(-\Delta f / t_k)$) **alors** $x_{i+1} = y$;
sinon $x_{i+1} = x_i$; }
$i = i+1$;
}
$t_{k+1} = g(t_k)$;
}
return x_{min} ;

MÉTHODE DU RECUIT SIMULÉ (SIMULATED ANNEALING)

- Exemple : **minimiser** la fitness f sur $\{0,1\}^4$

x	**0000**	**0001**	**0010**	**0011**	**0100**	**0101**	**0110**	**0111**
f(x)	12	5	9	2	11	3	8	7

x	**1000**	**1001**	**1010**	**1011**	**1100**	**1101**	**1110**	**1111**
f(x)	6	10	6	4	0	11	1	5

Transformation locale : « inverser un bit »

$t_0 = 4$

choix aléatoire

$x_0 = 0000$ $f(x_0) = 12$ $V(x_0) = \{ 1000, 0100, 0010, 0001 \}$
6 11 9 5

choix aléatoire

$x_1 = 0010$ $f(x_1) = 9$ $V(x_1) = \{ 1010, 0110, 0000, 0011 \}$
6 8 12 2

MÉTHODE DU RECUIT SIMULÉ (SIMULATED ANNEALING)

- Exemple : **minimiser** la fitness f sur $\{0,1\}^4$

x	0000	0001	0010	0011	0100	0101	0110	0111
f(x)	12	5	9	2	11	3	8	7

x	1000	1001	1010	1011	1100	1101	1110	1111
f(x)	6	10	6	4	0	11	1	5

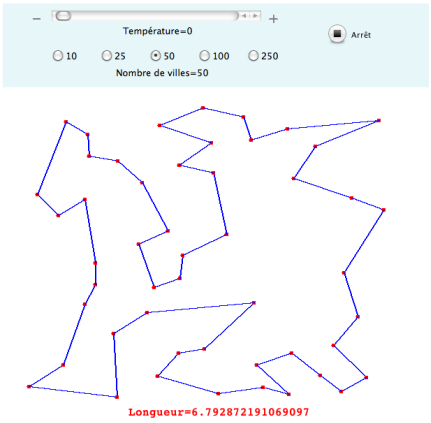
$x_2 = 0011 \quad f(x_2) = 2 \quad V(x_2) = \{ 1011, 0111, 0001, 0010 \}$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad 4 \quad 7 \quad 5 \quad 9$ ← choix aléatoire

Comme $f(0001) > f(x_2)$:

Tirer $p \in [0,1]$, | si $p < \exp(-3/4)$ alors $x_3 = 0001$
| sinon $x_3 = x_2$

MÉTHODE DU RECUIT SIMULÉ (SIMULATED ANNEALING)

- Exemple



http://interstices.info/jcms/c_43811/le-recuit-simule

MÉTHODE DU RECUIT SIMULÉ (SIMULATED ANNEALING)

- Paramétrage de l'algorithme : x_0, t_0, $g(.)$, n_1, n_2, …

  → une des difficultés des métaheuristiques

- Température initiale t_0 : suffisamment élevée pour accepter beaucoup de « moins bonnes solutions » :

  tester différentes solutions couteuses ($\Delta f>0$)

  fixer t_0 de manière à avoir 1 chance sur 2 d'accepter ces solutions :

  $$\exp(-\Delta f/t_0) = 0.5$$

  $\Rightarrow t_0 = -\Delta f/\ln(0.5)$

MÉTHODE DU RECUIT SIMULÉ (SIMULATED ANNEALING)

- Fonction de décroissance de la température :

  $$g(t) = \mu t \quad (0<\mu<1)$$

  des valeurs de μ proches de 1,

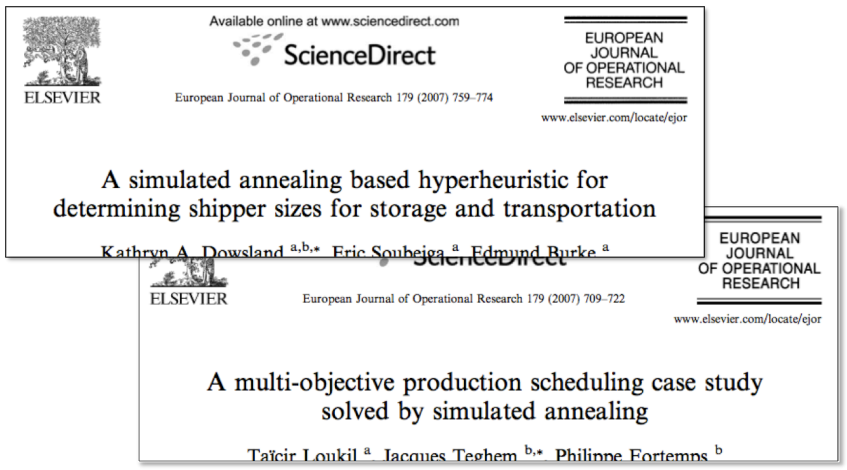
  éventuellement, faire varier μ au cours du temps :

  - décroissance élevée au début (μ faible),
  - décroissance lente ensuite (μ proche de 1),

- Nombre de changements de température n_1 :

  si μ=0.95 et n_1=60, alors une solution qui aurait une probabilité de 0.5 d'être acceptée au début, aura une probabilité de 3×10^{-7} de l'être après n_1 changements.

MÉTHODE DU RECUIT SIMULÉ (SIMULATED ANNEALING)



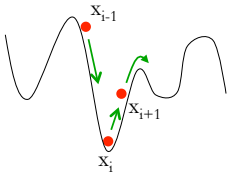
MÉTHODE TABOU

- **Méthode Tabou (Tabu search)**
  Liste taboue T, de taille finie, contenant les « solutions » (ou « directions » ou plutôt les « transformations inverses ») interdites.

Principe :

- à chaque x_i , choisir x_{i+1} qui minimise f(.) sur $V(x_i) - T$
- si $f(x_i)<f(x_{i+1})$, accepter x_{i+1} et s'interdire de redescendre sur x_i
  sinon, on boucle

si $x_{i+1}=m(x_i)$, alors on ajoute m^{-1} à T
→ transformation élémentaire
si T pleine, on supprime la transformation la plus ancienne



MÉTHODE TABOU

x_0 : solution initiale ; T=∅ ;
$x_{min}= x_0$; $f_{min}=f(x_{min})$; i=0 ;

répéter
{ C=V(x_i) – {m(x_i) / m∈T} ;
si (C≠∅) **alors**
{ Choisir y∈C tel que f(y)=min{ f(z) / z∈C } ; // y=m(x_i)
Δf = f(y) – f(x_i) ;
si (Δf ≥ 0) **alors** { mettre m^{-1} dans T ; }
si (f(y) < f_{min}) **alors** { f_{min}= f(y) ;
x_{min}= y ; }
x_{i+1} = y ;
}
i = i+1 ;
} **jusqu'à** ((i=nmax) ou (C=∅)) ;
return x_{min} ;

MÉTHODE TABOU

- Exemple : **minimiser** la fitness f sur $\{0,1\}^4$

x	0000	0001	0010	0011	0100	0101	0110	0111
f(x)	12	5	9	2	11	3	8	7

x	1000	1001	1010	1011	1100	1101	1110	1111
f(x)	6	10	6	4	0	11	1	5

Transformation locale : « inverser un bit » - Liste taboue de taille 1

x_0 = 0000 $f(x_0)$ = 12 $V(x_0)$ = { 1000, 0100, 0010, 0001 } T = ∅
6 11 9 5

x_1 = 0001 $f(x_1)$ = 5 $V(x_1)$ = { 1001, 0101, 0011, 0000 } T = ∅
6 3 2 12

x_2 = 0011 $f(x_2)$ = 2 $V(x_2)$ = { 1011, 0111, 0001, 0010 } T = ∅
4 7 5 9

MÉTHODE TABOU

- Exemple : **minimiser** la fitness f sur $\{0,1\}^4$

x	0000	0001	0010	0011	0100	0101	0110	0111
f(x)	12	5	9	2	11	3	8	7

x	1000	1001	1010	1011	1100	1101	1110	1111
f(x)	6	10	6	4	0	11	1	5

$x_3 = 1011 \quad f(x_3) = 4 \quad V(x_3) = \{ \sout{0011}, 1111, 1001, 1010 \} \quad T = \{m_1\}$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad 5 \quad 10 \quad 6$

$x_4 = 1111 \quad f(x_4) = 5 \quad V(x_4) = \{ 0111, \sout{1011}, 1101, 1110 \} \quad T = \{m_2\}$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad 7 \quad 11 \quad 1$

$x_5 = 1110 \quad f(x_5) = 1 \quad V(x_5) = \{ 0110, \sout{1010}, 1100, 1111 \} \quad T = \{m_2\}$
$\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\quad 8 \quad 0 \quad 5$

…

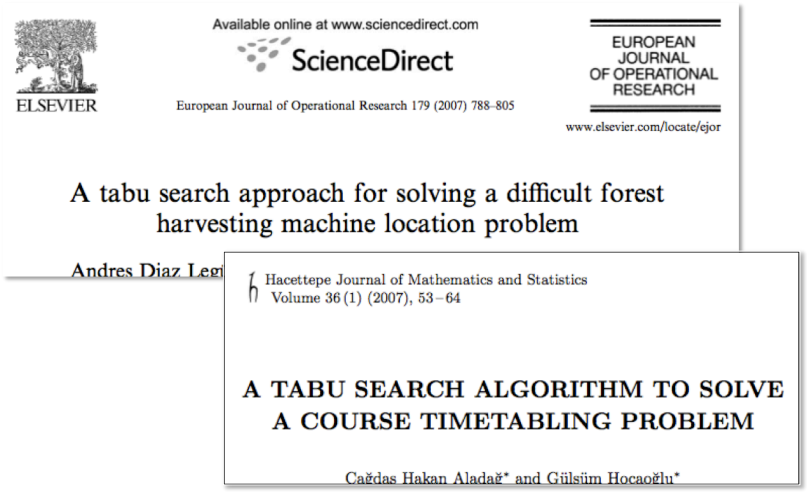
MÉTHODE TABOU

- recuit simulé : non déterministe, sans mémoire
  méthode Tabou : déterministe, avec mémoire

- Modifications possible de l'algorithme :

  1. il est possible d'interdire de « remonter » d'où on vient de « descendre »
  2. autoriser des transformations taboues qui améliorent le f_{min} (la meilleure solution rencontrée)

MÉTHODE TABOU



ELSEVIER

Available online at www.sciencedirect.com

ScienceDirect

European Journal of Operational Research 179 (2007) 788–805

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

www.elsevier.com/locate/ejor

A tabu search approach for solving a difficult forest harvesting machine location problem

Andres Diaz Legi

Hacettepe Journal of Mathematics and Statistics
Volume 36 (1) (2007), 53 – 64

A TABU SEARCH ALGORITHM TO SOLVE A COURSE TIMETABLING PROBLEM

Cağdas Hakan Aladağ* and Gülsüm Hocaoğlu*