

## CH.5 MÉTHODES APPROCHÉES

- 5.1 Les voisinages
- 5.2 Les méthodes de descentes
- 5.3 Le recuit simulé
- 5.4 La Méthode Tabou
- 5.5 Les algorithmes génétiques

### 5.1 Les voisinages

Dans les problèmes de complexité exponentielle, il est souvent hors de question de réaliser une exploration, même partielle, de l'arborescence des solutions.

Une stratégie consiste alors à partir d'une solution approchée et à la modifier localement (cas courant pour la confection d'emplois du temps).

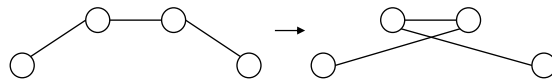
Il faut déterminer ce qu'on entend par modification locale. Cela dépend du problème, de son codage et de la facilité de réaliser cette modification.

Pour le problème du sac à dos, on peut envisager comme modification d'enlever un composant et de le remplacer par un autre, ou d'augmenter le nombre de composants d'un type donné, si la contrainte de volume continue à être satisfaite,

Dans notre exemple, maximiser  $10x_1 + 8x_2 + 5x_3$  avec les contraintes de positivité (sous-entendues) et la contrainte  $6x_1 + 5x_2 + 4x_3 \leq 9$ , en partant de la solution  $x_1 = 1, x_2 = x_3 = 0$ , on pourrait essayer  $x_1 = 0, x_2 = 0, x_3 = 1$ , puis  $x_1 = 0, x_2 = 1, x_3 = 1$ , qui par chance est la solution optimale.

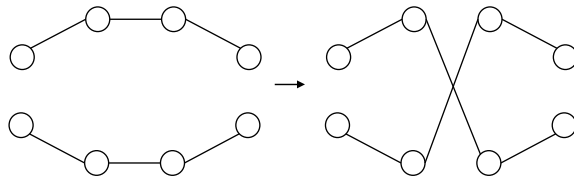
Dans le cas du voyageur de commerce, il faut envisager une opération qui conserve la propriété d'être un cycle hamiltonien. On peut échanger deux sommets consécutifs du cycle ou effectuer l'opération 2-opt de Lin et Kernighan qui agit sur deux couples de sommets consécutifs.

Echange de sommets consécutifs



Opti-comb ch 5 3

Opération 2-opt



Si le graphe comporte  $n$  sommets, il y a donc  $n$  opérations du type échange de sommets consécutifs et  $n(n-3)/2$  opérations 2-opt.

L'ensemble des solutions qui se déduisent d'une solution particulière par une modification locale est le *voisinage* de cette solution.

Cette notion dépend de l'ensemble des modifications locales acceptées.

Opti-comb ch 5 4

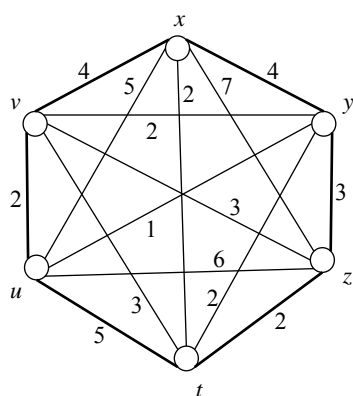
## 5.2 Les méthodes de descentes

Chaque modification fait varier le coût de la solution. Les algorithmes décrits par la suite consistent à partir d'une solution obtenue d'une manière ou d'une autre, à la modifier localement et à observer la variation du coût.

Partant d'une solution, on peut considérer toutes les modifications applicables et retenir la meilleure. On itère jusqu'à ce qu'aucune modification n'améliore le coût.

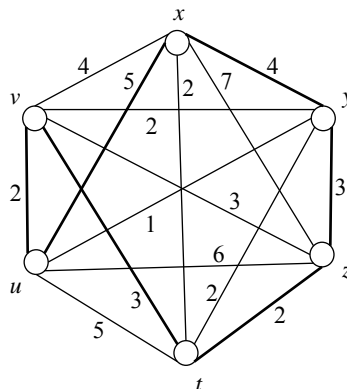
Par exemple, dans le cas du voyageur de commerce et de l'échange de sommets consécutifs, on trouverait, en partant du cycle  $xyztuv$  :

Opti-comb ch 5 5



Début : poids 20

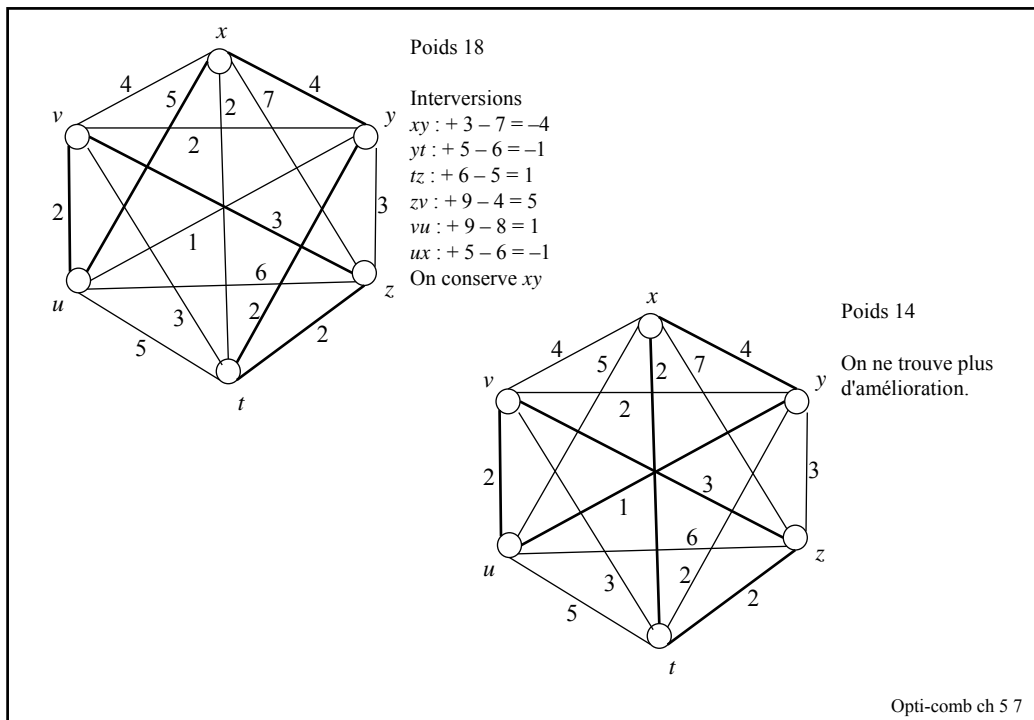
Interversions  
 $xy : +9 - 7 = 2$   
 $yz : +9 - 6 = 3$   
 $zt : +8 - 8 = 0$   
 $tu : +9 - 4 = 5$   
 $uv : +8 - 9 = -1$   
 $vx : +7 - 6 = 1$   
 On conserve  $uv$



Poids 19

Interversions  
 $xy : +8 - 8 = 0$   
 $yz : +9 - 6 = 3$   
 $zt : +5 - 6 = -1$   
 $tv : +8 - 4 = 4$   
 $vu : +9 - 8 = 1$   
 $ux : +5 - 6 = -1$   
 On conserve  $zt$

Opti-comb ch 5 6



Cette méthode peut donner de bons résultats lorsque le nombre de sommets est petit.

Mais on ne sait pas si on obtient une solution optimale et on ne sait pas quand : l'analyse en temps est délicate. Par ailleurs, si  $n$  est grand, l'examen systématique des  $n$  (ou plus avec l'opération 2-opt) possibilités peut trop ralentir l'algorithme.

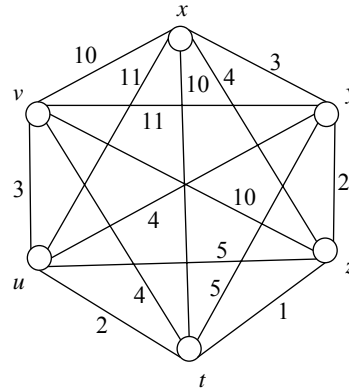
Ce dernier inconvénient peut être pallié en effectuant des tirages aléatoires de modifications et en ne retenant une modification que si elle est meilleure. Il faut alors prévoir un critère d'arrêt (nombre d'essais sans amélioration, par exemple).

### 5.3 Le recuit simulé

En fait, l'inconvénient de non-convergence vers l'optimum peut se rencontrer assez vite.

Par exemple, pour le graphe suivant, si on part du cycle hamiltonien  $xyztuv$ , aucune méthode de descente utilisant 2-opt ne conduit à la solution optimale.

Cela reste vrai quel que soit le sommet de départ si on initialise le cycle par un algorithme "glouton".

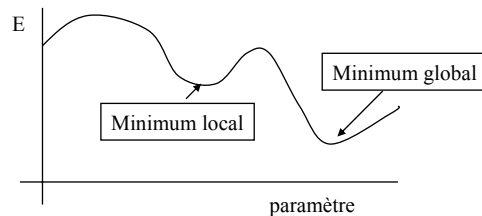


Au voisinage d'une solution obtenue, il n'existe pas de solution meilleure mais cette solution obtenue n'est pas la meilleure dans l'absolu.

Opti-comb ch 5 9

On se trouve dans un "puits de potentiel".

Un système physique se mettra de lui-même dans un état d'énergie *minimale*. Mais un tel état ne correspond pas nécessairement à un état d'énergie *minimum*.



Pour passer du minimum local au minimum global, il faut accepter de franchir une barrière en augmentant l'énergie temporairement.

Opti-comb ch 5 10

En chimie, un catalyseur permet de passer d'un état stable à un autre d'énergie inférieure en fournissant un moyen de franchir cette barrière de potentiel.

Dans un solide, les molécules sont plus ou moins mobiles en fonction de la température. Lorsque celle-ci diminue brusquement, les molécules se figent dans un état qui n'est pas nécessairement celui de plus petite énergie. En métallurgie, on appelle ce phénomène la *trempe*. Pour atteindre un état de plus faible énergie, donc un cristal ayant moins de défauts, il est nécessaire de réchauffer le métal, puis de le laisser se refroidir aussi lentement que possible. Cette technique est appelée le *recuit*. Idéalement, il faudrait que la température décroisse infiniment lentement. En fait, on la laisse décroître lentement, puis on rechauffe et ainsi de suite plusieurs fois. Le choix des températures successives de recuit et le temps qu'on laisse refroidir le métal font partie des secrets de fabrication assurant la qualité du métal produit...

Opti-comb ch 5 11

L'idée est de permettre des modifications locales susceptibles d'augmenter le coût (si cette fonction est à minimiser). Ces modifications vont peut-être permettre de franchir une barrière et aboutir à une solution meilleure.

Le processus de recuit simulé est contrôlé par différents paramètres. Les bonnes valeurs pour ces paramètres ne sont pas précisément connues. Le processus doit donc être évalué avant d'être utilisé à l'échelle réelle de façon à ajuster ces divers paramètres.

Supposons que le coût d'une configuration  $X_i$  soit  $f(X_i)$ . Une modification locale fait passer à une configuration  $X_j$  de coût  $f(X_j)$ . la décision de conserver  $X_j$  à la place de  $X_i$  est prise de la manière suivante :

Opti-comb ch 5 12

Soit  $\Delta f_{ij} = f(X_j) - f(X_i)$  ;  
 - si  $\Delta f_{ij} \leq 0$ , conserver  $X_j$  ;  
 - si  $\Delta f_{ij} > 0$ , conserver  $X_j$  avec probabilité  $\exp(-\Delta f_{ij} / T)$ , ou  $T$  est appelée *température*.

Cette distribution de probabilité est inspirée de la loi de physique statistique de *Gibbs-Boltzmann*. Dans le contexte de l'informatique, ce choix est connu sous le nom de *règle de Metropolis*.

On voit que, plus la température est élevée, plus la probabilité de conserver une configuration moins bonne est grande. Par contre, lorsque la température est nulle, seules les modifications faisant diminuer le coût seront conservées.

Un déroulement possible de l'algorithme est le suivant :

Opti-comb ch 5 13

```

 $X_{opt} = X ; f_{opt} = f(X_{opt}) ; T = T_0 ;$  /* initialisations */
itérer jusqu'à ce qu'on décide d'arrêter
{ itérer un nombre de fois palier fixé d'avance
  { choisir aléatoirement  $X'$  dans  $V(X)$  ;
    si  $\Delta f \leq 0$ , alors /* le coût diminue */
    {  $X = X'$  ;
      si  $f(X) < f(X_{opt})$ , alors /* on garde la meilleure */
      {  $X_{opt} = X ; f_{opt} = f(X_{opt}) ;$  /* configuration */
      }
    }
  }
  sinon /*  $\Delta f > 0$  */
  { tirer  $p$  au hasard entre 0 et 1 ;
    si  $p \leq \exp(-\Delta f / T)$ , alors  $X = X'$  ; /* on accepte  $X'$  */
  }
}
 $T = g(T) ;$  /* faire décroître la température */
}
retourner  $X_{opt}$ 

```

Opti-comb ch 5 14

On fait un nombre de modifications égales à une constante sans changer la température, puis on fait décroître celle-ci. La fonction  $g$  choisie est en général une suite géométrique décroissante de raison plus petite que 1 (essayer autour de 0,9).

La température initiale  $T_0$  est choisie telle qu'environ 50% à 60% des modifications sont acceptées. Mais on peut essayer diverses autres valeurs. Si on se fixe une proportion  $P$  de modifications à accepter, on engendre des modifications aléatoires et on calcule la variation du coût moyen  $\Delta f$ . On en déduit  $T_0 = -\Delta f / \ln P$ . C'est évidemment très dépendant de la qualité de la solution initiale.

Quant à la solution de départ, on utilise un algorithme quelconque pour la fabriquer.

Opti-comb ch 5 15

La solution optimale finit par se stabiliser. On arrête lorsqu'on n'observe plus d'amélioration de celle-ci pendant un temps assez long.

Enfin, la valeur du palier peut rester constante (en général dépendante de la taille du problème) ou décroître en même temps que la température.

Cette méthode a été introduite aux environs de 1982 par diverses personnes en s'inspirant de simulations numériques de l'évolution d'un système physique effectuées précédemment par Metropolis.

On trouve des exemples interactifs de cette méthode appliquée au voyageur de commerce aux adresses :

<http://www-sop.inria.fr/mefisto/java/tutorial1/node18.html>

ou

[http://interstices.info/jcms/c\\_43811/le-recuit-simule](http://interstices.info/jcms/c_43811/le-recuit-simule)

Opti-comb ch 5 16

Dans le cas du voyageur de commerce entre  $n$  points aléatoires d'un carré de côté 1, la valeur moyenne de la longueur d'un tel parcours est d'environ  $0,749 \sqrt{n}$ . Cette valeur est obtenue par des moyens probabilistes mais ne fournit pas de parcours correspondant.

On peut ainsi estimer la qualité du résultat obtenu par rapport à la valeur moyenne théorique.

#### **5.4 La Méthode Tabou**

On permet ici aussi d'examiner des solutions qui font augmenter le coût. L'algorithme est complètement déterministe (donc pas de tirage au sort). A chaque étape de l'algorithme, on examine la solution du voisinage de la solution courante qui minimise la fonction coût. Si la solution courante n'est pas un minimum local, on va rapidement trouver ce minimum. Si c'est déjà un minimum local, on s'intéresse à la solution qui fait augmenter le coût le moins possible.

On est sûr dans ce cas de tomber sur un cycle. Pour éviter cela, on garde à jour la liste des solutions déjà explorées (liste des solutions *tabou*), et on se limite aux solutions qui n'appartiennent pas à cette liste.

Cette liste peut devenir de taille trop grande. Plutôt que de retenir les solutions, on peut donc retenir les modifications locales ayant produit les solutions à partir de la solution initiale. On a donc une liste de modifications qu'on s'interdit de remonter. Pour éviter de bloquer complètement l'algorithme, il est bon de ne pas se souvenir de *toutes* les modifications depuis le début, mais seulement des dernières (de 3 à 15 environ), en les gérant sous forme d'une file.

Ceci dit, aucune garantie n'existe de convergence ni même de non-circularité. Il faut donc prévoir un test d'arrêt.

Lorsque l'algorithme s'arrête, on garde la meilleure solution rencontrée.

De nombreuses variantes ont été proposées, par exemple en autorisant des mouvements tabou lorsque l'amélioration du coût dépasse un certain seuil, ou bien en introduisant des choix aléatoires de mouvements, ...

Les résultats sont comparables à ceux utilisant un recuit simulé, parfois meilleurs, parfois moins bons. Aucune règle générale ne semble s'imposer...

Cette méthode a été introduite simultanément par plusieurs (P. Hansen, F. Glover, B. Jaumard) aux environs de 1986.

## 5.5 Les algorithmes génétiques

Les approches précédentes raisonnent sur une seule solution à la fois.

Au contraire, les algorithmes génétiques s'intéressent à un ensemble de solutions simultanément.

Par analogie avec la génétique, cet ensemble de solutions est une *population*. Chaque solution en est un *individu*. Chaque individu est codé de façon appropriée par son *génome*. La fonction coût, qu'on veut minimiser, sert d'estimateur de la qualité de chaque individu.

Les meilleurs individus sont autorisés à se *reproduire*. Pour cela, deux individus vont "croiser" leur génome au hasard opération dite de *crossing over*. Leur rejeton aura un génome composé de gènes de l'un ou l'autre de ses parents.

Opti-comb ch 5 21

Ce rejeton est ajouté à la population, dont on exclut l'individu de moins bonne qualité.

On choisit, proportionnellement à leur valeur dans la population (fonction à définir) deux reproducteurs, on les croise et on recommence.

On garde bien entendu trace du meilleur individu rencontré jusqu'à présent. On s'arrête quand on n'observe plus d'amélioration de la qualité de la population.

Pour introduire de la variabilité supplémentaire, on autorise un taux de "mutations", c'est-à-dire une modification d'un gène du génome d'un individu de temps en temps.

On a plus défini un principe qu'un algorithme. Le nombre de paramètres est en effet très grand.

Opti-comb ch 5 22

Il faut définir le génome d'un individu.  
Puis il faut définir les modalités de croisement des génomes.  
Enfin, il faut définir les mutations autorisées.

A chaque fois, des valeurs numériques sont à définir (taille de la population, sélection des individus aptes à se reproduire, nombre de générations, taux de mutation, critère d'arrêt).

Dans le cas du sac à dos, il y a autant de gènes que de types d'article.  
Le gène correspondant à un type d'article est le nombre d'articles de ce type. Le génome est donc le vecteur donnant le nombre de chaque article. Pour le rejeton, le nombre d'articles de chaque type est celui de l'un ou l'autre de ses parents. Certains individus ne seront pas viables (si la contrainte de volume n'est pas satisfaite). Ils seront donc rejetés.

La mutation peut être, si c'est possible, un échange d'une unité entre deux gènes voisins.

Opti-comb ch 5 23

Par, pour maximiser  $10x_1 + 8x_2 + 5x_3$  avec la contrainte  $6x_1 + 5x_2 + 4x_3 \leq 30$ , les individus  $(3, 1, 0)$  et  $(2, 2, 1)$  pourraient, en se croisant, donner l'individu  $(3, 2, 1)$ , non viable ( $V = 32$ ) ou l'individu  $(3, 1, 1)$ , viable ( $V = 27$ ). Ce dernier, à son tour, pourrait muter en  $(3, 2, 0)$ , ...

Mais d'autres règles de croisement et de mutation sont envisageables.

En ce qui concerne le voyageur de commerce, le génome semble être la permutation circulaire des sommets parcourus en partant d'un sommet fixé une fois pour toutes. Avec  $n$  sommets, on a ainsi un  $(n - 1)!$  génomes différents.

Le croisement est un peu délicat. On peut chercher un cycle qui fait passer d'une permutation à l'autre, puis conserver les éléments de ce cycle d'une des permutations et compléter par les éléments restants de l'autre :

Opti-comb ch 5 24

Pour croiser (**1 6 5 2 4 3**) et (**3 1 2 4 5 6**), on trouve le cycle suivant 1 – 3, puis 3 – 6 enfin 6 – 1 qui termine le cycle (en gras sur les permutations), ce qui fait comme croisement (**1 6 2 4 5 3**). (*Cycle crossover*)

Une mutation serait alors un échange de deux sommets consécutifs.

On peut aussi utiliser un codage de Lehmer : le  $i$ -ième gène d'une permutation chaque permutation est le nombre d'éléments de la permutation situés *dans les  $i - 1$  premières places et plus petits que le  $i$ -ième*. Le génome de (1 6 5 2 4 3) est (0 1 1 1 2 2), et celui de (3 1 2 4 5 6) est (0 0 1 3 4 5). Chaque gène du fils est alors pris chez l'un ou l'autre parent, aléatoirement ; par exemple (0 0 1 1 2 5), qui correspond à la permutation (5 1 4 2 3 6)... La mutation consiste à modifier un élément du code d'une unité (ce qui revient d'ailleurs à permuter deux sommets consécutifs du chemin...)

Opti-comb ch 5 25

Le reste est affaire d'expérimentation pour trouver les meilleurs gènes et les meilleurs paramètres. Il est préférable que les solutions proches de l'optimum aient un pouvoir de reproduction un peu moindre, pour éviter de perdre de bonnes combinaisons de gènes...

Ce type d'algorithme est apparu dans les années 60, introduits par J.H. Holland à l'université du Michigan dans l'étude de systèmes adaptatifs. Il a introduit un cadre théorique à ces algorithmes (la *théorie des schémas*).

Une démonstration d'algorithme génétique pour le voyageur de commerce peut-être trouvée sur

<http://magnin.plil.net/anciensite/coursag/voyageur/voyageur.html>

Une autre application d'algorithme génétique se trouve sur

<http://www.rennard.org/alife/french/gav.html>

Opti-comb ch 5 26