République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

N° Réf :……

Centre Universitaire de Mila

Institut des Sciences et de la Technologie **Département de Mathématiques et Informatique**

Mémoire préparé En vue de l'obtention du diplôme de licence

En : - Filière : Mathématiques Fondamentales

Thème : Récuit simulé

- Préparé par : - Hézili Sarra .
- Lemiz Wafia.
- Slimani Soumia.
- Serikma Rafika.

- Encadré par : Zaidi Ali

Année universitaire : 2013/2014

REMERCÎMENTS

Nous remercions notre encadreur Ali Zaidi pour la proposition du thème, pour ces conseils prodigues lors de la rédaction du mémoire et bien sur pour l'encadrement.

Nous remercions également tous les enseignants du département de mathématique du centre universitaire de Mila , qui nous ont instruit tout au long de notre trajet de licence L.M.D.

Tables des matieres

1. Introduction :

L'optimisation est sans aucun doute un domaine de recherche très important pour les scientifiques et les ingénieurs et elle est aussi une branche de mathématique cherchant à analyser et a résoudre des problèmes de meilleurs éléments (déversions) d'un ensemble afin de maximiser ou de minimiser un critère qualitatif qui mesure la qualité de la décision.

Le mot optimisation vient du latin optimum : qui signifie « le meilleur » l'optimisation un rôle important en économies Finance, où analyse numérique, en statistique (l'ensemble de vraisemblance d'une distribution) la recherche de stratégie dans le cache de la théorie des jeux et aussi en théorie de contrôle.

L'optimisation combinatoire occupe une place très importante en recherche opérationnelle .Son importance se justifie d'une part par la grande difficulté des problèmes d'optimisation et d'autre part.par les nombreuses applications pratiques pouvant être formulées sous la forme d'un problème d'optimisation combinatoire.

Bien que les problèmes d'optimisation combinatoire soient souvent faciles à résoudre.

En fait, la plupart de ces problèmes NP e et ne possèdent donc pas à ce jour de solution.

2. Optimisation combinatoire

Un problème d'optimisation combinatoire est défini par un ensemble d'instances.
A chaque instance du problème est associé un ensemble discret de solutions S, un sous-ensemble X de S représente les solutions admissibles (réalisables) et une fonction de coût f (ou fonction objectif) qui assigne à chaque solution $S \in X$ le nombre réel (ou entier) $f(S)$. Résoudre un tel problème (plus précisément une telle instance du problème) consiste à trouver une solution $S^* \in X$ optimisant la valeur de la fonction de coût f. Une telle solution s^* s'appelle une *solution optimale* ou un *optimum global*.

Nous avons donc la définition suivante :
Définition [3]. Une *instance I* d'un problème de minimisation est un couple (X, f) où $X \subseteq S$ est un ensemble fini de solutions admissibles, et f une fonction de coût (ou objectif) à minimiser $f : X \rightarrow R$. Le problème est de trouver $S^* \in X$ tel que $f(S^*) \leq f(S)$ pour tout élément $S \in X$.

Notons que d'une manière similaire, on peut également définir les problèmes de maximisation en remplaçant simplement « par ». L'optimisation combinatoire trouve des applications dans des domaines aussi variés que la gestion, l'ingénierie, la conception, la production, les télécommunications, les transports, l'énergie, les sciences sociales et l'informatique elle-même**.**

3.Les problèmes classiques d'optimisation combinatoire

3.1- **voyageur de commerce** :

Dans lequel un représentant doit visiter un certain nombre de villes, avant de retourner à son point de départ : la question est de déterminer le trajet le plus court traversant chaque ville une seule fois, c'est-à-dire la succession de villes qui minimise la tournée du représentant.

De nombreux problèmes d'ingénierie peuvent se ramener au problème du voyageur de commerce (PVC), d'où son intérêt. On le retrouve dans le domaine des réseaux informatiques par exemple, avec les algorithmes de routage.

Par ailleurs, le PVC est un cas particulier d'un problème plus général, celui des tournées de véhicules, qui consiste à calculer les meilleurs parcours pour une flotte de véhicules devant livrer un ensemble de produits dans plusieurs destinations à partir d'un entrepôt. Chaque véhicule a une capacité limitée. Le problème des tournées de véhicules trouve ses applications dans le domaine de la distribution bien sûr, mais aussi dans les télécoms.

3.2-Problème d'ensembles indépendants

Ce problème d'optimisation combinatoire se rencontre chaque fois qu'il s'agit de minimiser une fonction de coût en présence de contraintes rigides. On peut formaliser le problème sous la forme suivante:

Soit un ensemble de configurations et un ensemble de configurations réalisables (satisfaisant les contraintes).

On introduit une fonction, pour définie par :

pour tout. Le paramètre est un facteur de pénalité. Les solutions réalisables contribuent uniquement au premier terme de la somme -- en l'abaissant -- tandis que les configurations irréalisables contribuent au premier et au second terme. Pour trouver donc l'ensemble indépendant maximal on peut utiliser l'algorithme du recuit simulé pour minimiser la fonction .La procédure suivante donne un mouvement isotherme de l'algorithme de minimisation

3.3- **l'affectation quadratique** :

Un autre problème classique est celui de : qui, par rapport au PVC, introduit la notion de *flot* circulant entre les villes, et qui peut s'énoncer ainsi : il s'agit de déterminer comment placer *n* objets dans *n* emplacements de manière à minimiser la somme des produits *flots* par *distances*. Mathématiquement, cela revient à minimiser :

$\sum_{i=1}^{n}\sum_{j=1}^{n} f_{ij} \cdot d_{ij}$ Où f_{ij} représente le flot entre l'objet *i* et l'objet *j*, et d_{ij} la distance entre l'objet *i* et l'objet *j*.

La répartition de bâtiments dans un espace donné en fonction du nombre de personnes amenées à circuler entre ces bâtiments, ou la façon de répartir des modules électroniques sur une carte en fonction du nombre de connexions les liant les uns aux autres, reviennent à résoudre le problème de l'affectation quadratique.

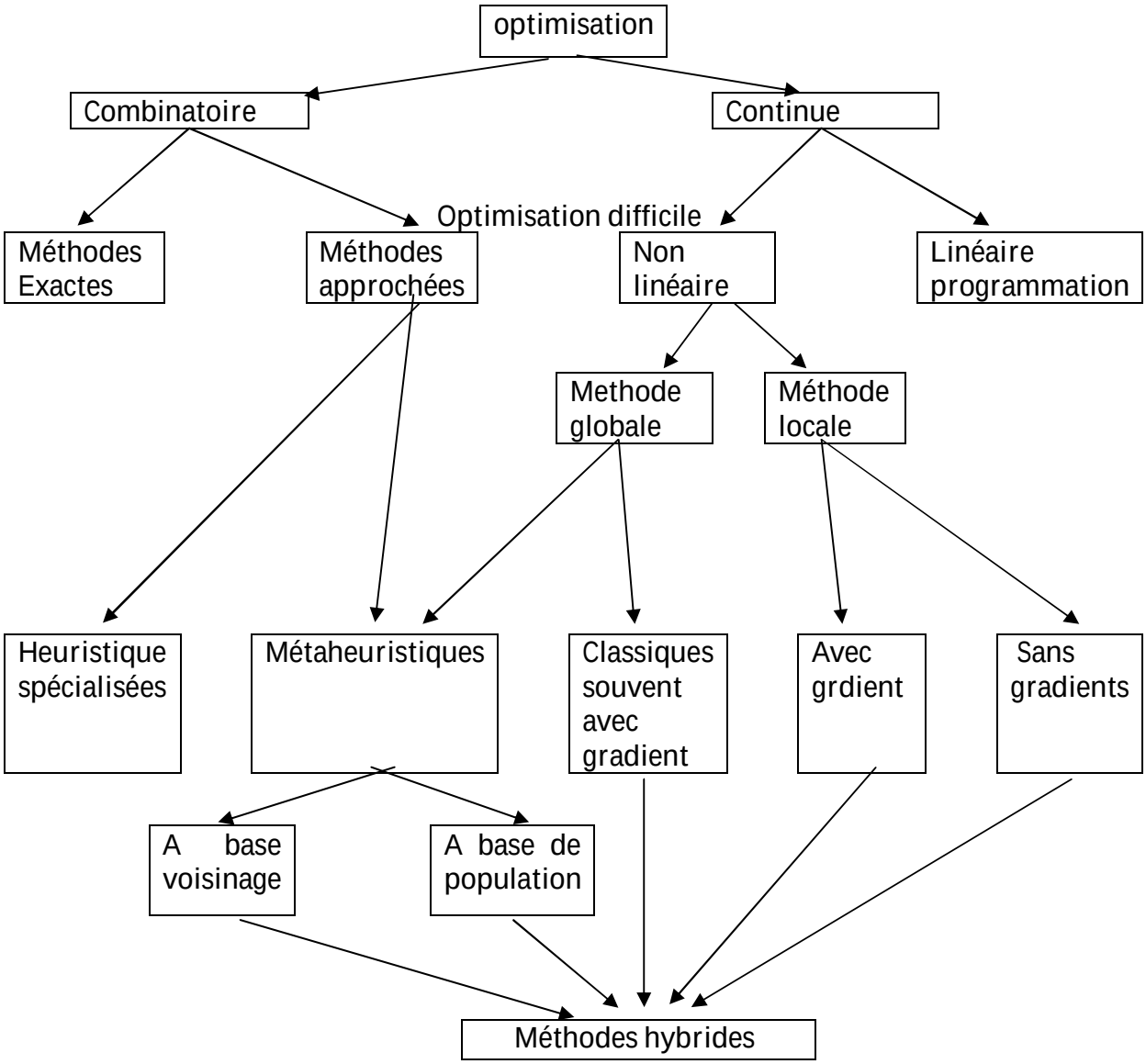
3.4-**l'ordonnancement de tâches** :

Dont l'énoncé classique est le suivant : soit un ensemble *m* de machines, un ensemble *t* de tâches à réaliser, chaque tâche étant composée d'une certaine suite d'opérations. Une machine ne peut réaliser qu'une seule opération à la fois.

Comment ordonner l'exécution des opérations sur les différentes machines de telle façon que le temps mis à faire tout le travail soit fait soit minimal ?

Naturellement, dans la vie courante, les situations sont encore plus compliquées, et il faut rajouter à ces situations-types des contraintes, des particularités, certaines « règles du jeux » qui viennent encore compliquer l'énoncé du problème à résoudre**.** [1]

4. Classification des méthodes d'optimisation mono-objectif



5-Les Méthodes de résolution exactes

Nous présentons d'abord quelques méthodes de la classe des algorithmes complets ou exacts, ces méthodes donnent une garantie de trouver la solution optimale pour une instance de taille finie dans un temps limité et de prouver son optimalité (Puchinger et Raidl 2005).

5.1.La Méthode séparation et évaluation (Branch and Bound)

L'algorithme de séparation et évaluation, plus connu sous son appellation anglaise Branch and Bound (B&B) (Land et Doig 1960), repose sur une méthode arborescente de recherche d'une solution optimale par séparations et évaluations, en représentant les états solutions par un arbre d'états, avec des nœuds, et des feuilles .Le branch-and-bound est basé sur trois axes principaux :

– L'évaluation
– La séparation
– La stratégie de parcours.

- L'évaluation

L'évaluation permet de réduire l'espace de recherche en éliminant quelques sous ensembles qui ne contiennent pas la solution optimale.

L'objectif est d'essayer d'évaluer l'intérêt de l'exploration d'un sous-ensemble de l'arborescence.

Le branch-and-bound utilise une élimination de branches dans l'arborescence de recherche de la manière suivante : la recherche d'une solution de coût minimal, consiste à mémoriser la solution de plus bas coût rencontré pendant l'exploration, et à comparer le coût de chaque nœud parcouru à celui de la meilleure solution. Si le coût du nœud considéré est supérieur au meilleur coût, on arrête l'exploration de la branche et toutes les solutions de cette branche seront nécessairement de coût plus élevé que la meilleure solution déjà trouvée.

- La séparation

La séparation consiste à diviser le problème en sous-problèmes.

Ainsi ,en résolvant tous les sous-problèmes et en gardant la meilleure solution trouvée, on est assuré d'avoir résolu le problème initial.

Cela revient à construire un arbre permettant d'énumérer toutes les solutions. L'ensemble de nœud de l'arbre qu'il reste encore à parcourir comme étant susceptibles de contenir une solution optimale, c'est-à-dire encore à diviser, est appelé ensemble des noeuds actifs.

- La stratégie de parcours

La largeur d'abord :

Cette stratégie favorise les sommets les plus proches de la racine en faisant moins de séparations du problème initial. Elle est moins efficace que les deux autres stratégies présentées.

La profondeur d'abord :

Cette stratégie avantage les sommets les plus éloignés de la racine (de profondeur la plus élevée) en appliquant plus de séparations au problème initial. Cette voie mène rapidement à une solution optimale en économisant la mémoire.

Le meilleur d'abord :

Cette stratégie consiste à explorer des sous problèmes possédant la meilleure borne. Elle permet aussi d'éviter l'exploration de tous les sous-problèmes qui possèdent une mauvaise évaluation par rapport à la valeur optimale.

5.2 La méthode de coupes planes (Cutting-Plane)

La méthode de coupes planes a été développée par (Schrijver 1986), elle est destinée à résoudre des problèmes d'optimisation combinatoire (POC) qui se formulent sous la forme d'un programme linéaire (PL) :

$\text{Min}\{c^T x : Ax \geq b , x \in R^n\}$ (1,1)

Dans le cas, où (POC) est de grande taille pour le représenter explicitement en mémoire ou pour qu'il tient dans un solveur de programmation linéaire, on utilise une technique qui consiste à enlever une partie de ces contraintes et de résoudre le problème relaxé (POCR).

La solution optimale de (PL) est contenue dans l'ensemble de solutions réalisables de cette relaxation. Pour un problème de minimisation la solution optimale du problème (POCR) est inférieure ou égale à la solution optimale donnée par (POC).

Cette méthode consiste à résoudre un problème relaxé, et à ajouter itérativement des contraintes du problème initial. On définit une contrainte pour le problème de minimisation (1.1) par le couple (s, s_0) où $s \in R^n$ et $s_0 \in R$, cette contrainte est dite violée par la solution courante x si pour tout $y \in \{ x : Ax > b\}$ on a : $s^T x < s_0$ et $s^T y \geq s_0$, on appelle alors ces contraintes des coupes planes. On arrête l'algorithme lorsqu'il n'y a plus de contraintes violées par la solution courante, on obtient ainsi une solution optimale pour le problème initial.

La méthode des coupes planes est peu performante mais sa performance est améliorée lorsqu'elle est combinée avec la méthode "Branch and Bound".**[2]**

5.3 Méthode des coupes de Gomory :

- Principe des méthodes de coupes Introduire de nouvelles contraintes linéaires au problème pour réduire le domaine réalisable du problème relaxé sans pour autant éliminer de points du domaine réalisable du problème avec les contraintes de nombre entier sur les variables.
- La procédure consiste à résoudre une suite de problèmes relaxés jusqu'à ce qu'une solution optimale en nombres entiers soit obtenue.
- Un problème de la suite est obtenu du précédent en lui ajoutant une contrainte linéaire (coupe) supplémentaire .le problème de programmation linéaire en nombres entiers suivant :

  (p) Min $\sum_{j=1}^{n} c_j x_j$

  Sujet à $\sum_{j=1}^{n} a_{ij} x_j = b_i$ $i = 1,2, \ldots . m$

  $x_j \geq 0$, $entier$, $j = 1 \ldots . n$

  la ligne correspondante du tableau est de la forme

  $x_k + \sum_{j \in j} t_{ij} x_j = \bar{b_i}$ (1)

où J={j :j est l'indice d'une variable hors base} et $\bar{b_i}$ n'est pas entier

Définition :[d] la plus grand entier (plan cher)≤d puisque $x_j \geq 0 \quad \forall j$ alors

$\sum_{j \in J} |t_{ij} x_j| \leq \sum_{j \in J} t_{ij} x_j$ et par conséquent $x_k + \sum_{j \in J} |t_{ij}| x_j \leq b_i$ (2)

Si nous considérons la contrainte d'intégralité des variables x_j ,il découle de (2) que

$X_k = \sum_{j \in J} |t_{ij}| x_j \leq |\bar{b_i}|$ (3)

Ainsi toute solution (P) est satisfis (P) est satisfait (3)
Considérons maintenons la relation option obtenue sur un faisant le différence entre (3) et (1) $\sum_{j \in J} (|t_{ij}| - t_{ij}) x_j \leq (|\bar{b_i}| - \bar{b_i})$ (4)
Notons que $(|t_{ij}| - t_{ij}) \leq 0$ et $|\bar{b_i}| - \bar{b_i} \leq 0$

Puisque toute solution de (P) satisfait (1) et (3), alors elle satisfait (4) et son introduction (P) n'élimine aucune solution (P).

Par contre, la solution actuelle du problème relaxé ($\bar{p}$) où $x_j = 0 \; \forall \; j \in J$ ne satisfait pas (4) et son introduction réduit le domaine réalisable du probléme relaxé (P) pour suivre la résolution,il suffit d'introduire la contrainte

$\sum_{j\in J}(|t_{ij}|-t_{ij})\ x_j \leq (|\bar{b}_i|-\bar{b}_i)+ x_t=(|\bar{b}_i|-\bar{b}_i)$ où x_t est une variable d' avec coût nul, au dernier tableau du simplexe pour générer une solution base de nouveau problème en considérant x_t comme la variable de base dans le nouvelle ligne du tableau .

Cette solution de base n'est pas réalisable puisque $x_t=(|\bar{b}_i|-\bar{b}_i)\leq 0$.

Il suffit de suivre la résolution avec l'algorithme dual du simplexe .

Notes

1) Si $|t_{ij}| = t_{ij}$(i.e., t_{ij})est entier $\forall$ j$\in$J , alors indique que (P) n'est pas réalisable puisque le terme de gauche prend une valeur entier pour toute solution réalisable de (P) alors que le terme de droite n'est pas entier.
2)Une dérivation similaire s'applique à toute les itération

Considérons le problème suivant :

Min $-21x_1-11x_2$

Sujet à $7x_1+4x_2+x_3$

$x_1, x_2, x_3 \geq 0$ entier

Itération 1 :

Solution optimal de $(\bar{p})$ de base

$x_1+(\frac{4}{7})\ x_2+(\frac{1}{7})\ x_3=(\frac{13}{7})$

Valeur opt=-39

Nouvelle contrainte

$[|(\frac{4}{7})|-(\frac{4}{7})\ x_2+[|(\frac{1}{7})|-(\frac{1}{7})\ x_3+x_4=|(\frac{13}{7})-(\frac{13}{7})|$

Interprétions géométrique :

$-(\frac{4}{7})\ x_2-(\frac{1}{7})\ x_3 \leq -(\frac{6}{7}) \Leftrightarrow -4x_2-x_3 \leq -6$

Or $x_3=13-7x_1-4x_2$ Ainsi $-4x_2-13+7x_1+4x_2 \leq -6 \Leftrightarrow x_1 \leq 1$

Itération 2

Résoudre le problème relaxé de Min $-21x_1-11x_2$

Sujet à $7x_1+4x_2+x_3=13$

$-(\frac{4}{7})\ x_2-(\frac{1}{7})\ x_3+x_4=-(\frac{6}{7})$

$x_1, x_2, x_3, x_4 \geq 0$ entier

Nous obtenons $x_1+x_4=1$

$x_2+(\frac{1}{4})\ x_3-(\frac{7}{4})x_4=(\frac{3}{2})$

Valeur opt=$-37(\frac{1}{2})$

Interprétation géométrique $-x_3-x_4 \leq -2$

Substituons le valeur de x4 tirée de $(|(\frac{4}{7})|-(\frac{4}{7}))\ x_1+(|(\frac{1}{7})|-(\frac{1}{7}))x_3+x_4-|(\frac{13}{7})|-(\frac{13}{7})$

Pour obtenir

$-x_3-(\frac{4}{7})\ x_2-(\frac{1}{7})\ x_3+(\frac{6}{7}) \leq -2 \Leftrightarrow 2x_1+x_2 \leq 3$ **[3]**

6.Métaheuristiques

Le mot métaheuristique est dérivé de la composition de deux mots grecs:
- heuristique qui vient du verbe heuriskein (euriskein) et qui signifie 'trouver'
- Méta qui est un suffixe signifiant 'au-delà', 'dans un niveau supérieur'.

On peut dire que les propriétés fondamentales des métaheuristiques sont les suivantes :

- Les métaheuristiques sont des stratégies qui permettent de guider la recherche d'une solution optimale
- Le but visé par les métaheuristiques est d'explorer l'espace de recherche efficacement afin de déterminer des solutions (presque) optimales.
- Les techniques qui constituent des algorithmes de type métaheuristique vont de la simple procédure de recherche locale à des processus d'apprentissage complexes.
- Les métaheuristiques sont en général non-déterministes et ne donnent aucune garantie d'optimalité.
- Les métaheuristiques peuvent contenir des mécanismes qui permettent d'éviter d'être bloqué dans des régions de l'espace de recherche.
- Les concepts de base des métaheuristiques peuvent être décrits de manière abstraite, sans faire appel à un problème spécifique.
- Les métaheuristiques peuvent faire appel à des heuristiques qui tiennent compte de la spécificité du problème traité, mais ces heuristiques sont contrôlées par une stratégie de niveau supérieur.
- Les métaheuristiques peuvent faire usage de l'expérience accumulée durant la recherche de l'optimum, pour mieux guider la suite du processus de recherche Les métaheuristiques sacrifient la garantie d'optimalité ou d'approximation avec en contrepartie l'espoir de trouver

très rapidement de bonnes solutions dans S.

Classifications possibles des Métaheuristiques

On peut faire la différence entre les métaheuristiques qui s'inspirent de phénomènes naturels et celles qui ne s'en inspirent pas. Par exemple, les algorithmes génétiques et les algorithmes des fourmis s'inspirent respectivement de la théorie de l'évolution et du comportement de fourmis à la recherche de nourriture. Par contre, la méthode Tabou n'a semble-t-il pas été inspirée par un phénomène naturel. Une telle classification ne semble cependant pas très utile est parfois difficile à réaliser. En effet, il existe de nombreuses métaheuristiques récentes qu'il est difficile de classer dans l'une des 2 catégories. Certains se demanderont par exemple si l'utilisation d'une mémoire dans la méthode Tabou n'est pas directement inspirée de la nature.

Une autre façon de classifier les métaheuristiques est de distinguer celles qui travaillent avec une population de solutions de celles qui ne manipulent qu'une seule solution à la fois. Les méthodes qui tentent itérativement d'améliorer une solution sont appelées méthodes de recherche locale ou méthodes de trajectoire. La méthode Tabou, le Recuit Simulé et la Recherche à Voisinages Variables sont des exemples typiques de méthodes de trajectoire. Ces méthodes construisent une trajectoire dans l'espace des solutions en tentant de se diriger vers des solutions optimales. L'exemple le plus connu de méthode qui travaille avec une population de solutions est l'algorithme génétique.

Les métaheuristiques peuvent également être classées selon leur manière d'utiliser la fonction objective. Étant donné un problème d'optimisation consistant à minimiser une fonction f sur un espace S de solutions, certaines métaheuristiques dites statiques travaillent directement sur f alors que d'autres, dites dynamiques, font usage d'une fonction g obtenue à partir de f en ajoutant quelques composantes qui permettent de modifier la topologie de l'espace des solutions, ces composantes additionnelles pouvant varier durant le processus de recherche.

Des chercheurs préfèrent classifier les métaheuristiques en fonction du nombre de structures de voisinages utilisées.

Étant donné qu'un minimum local relativement à un type de voisinage ne l'est pas forcément pour un autre type de voisinage, il peut être intéressant d'utiliser des métaheuristiques basées sur plusieurs types de voisinages.

Certaines métaheuristiques font usage de l'historique de la recherche au cours de l'optimisation, alors que d'autres n'ont aucune mémoire du passé. Les algorithmes sans mémoire sont en fait des processus markoviens puisque l'action à réaliser est totalement déterminée par la situation courante.

Les métaheuristiques qui font usage de l'historique de la recherche peuvent le faire de diverses manières. On différentie généralement les méthodes ayant une mémoire à court terme de celles qui ont une mémoire à long terme**.**

Finalement, mentionnons que certaines métaheuristiques utilisent les concepts additionnels que sont la diversification et l'intensification.

Par diversification, on sous-entend généralement une exploration assez large de l'espace de recherche, alors que le terme intensification vient plutôt mettre l'accent sur l'exploitation de l'information accumulée durant la recherche. Il est important de bien doser l'usage de ces deux ingrédients afin que l'exploration puisse rapidement identifier des régions de l'espace de recherche qui contiennent des solutions de bonne qualité, sans perdre trop de temps à exploiter des régions moins prometteuses.

Dans notre description des principales métaheuristiques, nous allons nous appuyer sur la classification qui distingue les méthodes de trajectoire des méthodes basées sur des populations de solutions.

6. *Le recuit simulé :*

Le recuit simulé est une méthode empirique (métaheuristique) inspirée d'un processus utilisé en métallurgie. On alterne dans cette dernière des cycles de refroidissement lent et de réchauffage (*recuit*) qui ont pour effet de minimiser l'énergie du matériau. Cette méthode est transposée en optimisation pour trouver les extrema d'une fonction.

Elle a été mise au point par trois chercheurs de la société IBM, S. Kirkpatrick, C.D. Gelatt et M.P. Vecchi en 1983, et indépendamment par V. Černy en 1985.

La méthode vient du constat que le refroidissement naturel de certains métaux ne permet pas aux atomes de se placer dans la configuration la plus solide. La configuration la plus stable est atteinte en maîtrisant le refroidissement et en le ralentissant par un apport de chaleur externe, ou bien par une isolation.

7.1-Déroulement du processus

Le recuit simulé s'appuie sur l'algorithme de Metropolis-Hastings, qui permet de décrire l'évolution d'un système thermodynamique. Par analogie avec le processus physique, la fonction à minimiser deviendra l'énergie du système. On introduit également un paramètre fictif, la température du système.

Partant d'une solution donnée, en la modifiant, on en obtient une seconde. Soit celle-ci améliore le critère que l'on cherche à optimiser, on dit alors qu'on a fait baisser l'énergie du système, soit celle-ci le dégrade.

Si on accepte une solution améliorant le critère, on tend ainsi à chercher l'optimum dans le voisinage de la solution de départ. L'acceptation d'une « mauvaise » solution permet alors d'explorer une plus grande partie de l'espace de solution et tend à éviter de s'enfermer trop vite dans la recherche d'un optimum local.

7.2-État initial de l'algorithme

La solution initiale peut être prise au hasard dans l'espace des solutions possibles.

À cette solution correspond une énergie initiale. Cette énergie est calculée en fonction du critère que l'on cherche à optimiser. Une température initiale élevée est également choisie. Ce choix est alors totalement arbitraire et va dépendre de la loi de décroissance

7.3-Itérations de l'algorithme

À chaque itération de l'algorithme une modification élémentaire de la solution est effectuée. Cette modification entraîne une variation de l'énergie du système (toujours calculée à partir du critère que l'on cherche à optimiser). Si cette variation est négative (c'est-à-dire qu'elle fait baisser l'énergie du système), elle est appliquée à la solution courante. Sinon, elle est acceptée avec une probabilité. Ce choix de l'exponentielle pour la probabilité s'appelle règle de Metropolis.

On itère ensuite selon ce procédé en gardant la température constante.

7.4-Programme de recuit

Deux approches sont possibles quant à la variation de la température :

1. Pour la première, on itère en gardant la température constante. Lorsque le système a atteint un équilibre thermodynamique (au bout d'un certain nombre de changements), on diminue la température du système. On parle alors de *paliers* de température.
2. La seconde approche fait baisser la température de façon continue. On peut imaginer toute sorte de loi de décroissance, la plus courante étant avec (assez couramment).

Dans les deux cas, si la température a atteint un seuil assez bas fixé au préalable ou que le système devient figé, l'algorithme s'arrête.

La température joue un rôle important. À haute température, le système est libre de se déplacer dans l'espace des solutions (proche de 1) en choisissant des solutions ne minimisant pas forcément l'énergie du système. À basse température, les modifications baissant l'énergie du système sont choisies, mais d'autres peuvent être acceptées, empêchant ainsi l'algorithme de tomber dans un minimum local.

7.5-Pseudo-code

Le pseudo-code suivant met en œuvre le recuit simulé tel que décrit plus haut, en commençant à l'état *s0* et continuant jusqu'à un maximum de *kmax* étapes ou jusqu'à ce qu'un état ayant pour énergie *emax* ou moins soit trouvé.

L'appel voisin(*s*) génère un état voisin aléatoire d'un état *s*. L'appel aléatoire() renvoie une valeur aléatoire dans l'intervalle [0, 1]. L'appel temp(r) renvoie la température à utiliser selon la fraction *r* du temps total déjà dépensé.

```
s := s0
e := E(s)
k := 0
tant que k < kmax et e > emax
  sn := voisin(s)
  en := E(sn)
  si en < e ou aléatoire() < P(en - e, temp(k/kmax)) alors
    s := sn; e := en
  k := k + 1
retourne s
```

7.6-Inconvénients

Les principaux inconvénients du recuit simulé résident dans le choix des nombreux paramètres, tels que la température initiale, la loi de décroissance de la température, les critères d'arrêt ou la longueur des paliers de température. Ces paramètres sont souvent choisis de manière empirique.

7.7-Étude théorique

Des études théoriques du recuit simulé ont pu montrer que sous certaines conditions, l'algorithme du recuit convergeait vers un optimum global. Ce résultat est important car il nous assure, contrairement à d'autres métaheuristiques, que le recuit simulé peut trouver la meilleure solution, si on le laisse chercher indéfiniment.

Les preuves de convergence reposent sur le principe de construction de l'algorithme du recuit simulé :

- Sous réserve qu'elle respecte la condition de Doeblin une chaîne de Markov admet une unique distribution stationnaire

- L'algorithme de Metropolis-Hastings permet, étant donnée une distribution de probabilité, de construire une chaîne de Markov dont la distribution stationnaire est cette distribution de probabilité.

Par ailleurs, les mesures de Gibbs, très utilisées en physique statistique, sont une famille de mesures dépendant d'un paramètre température de telle sorte que lorsque la température tend vers 0, la mesure converge vers un Dirac en un point donné (état d'énergie minimale).

En composant les deux, on obtient une chaîne de Markov qui converge vers une mesure invariante qui évolue dans le même temps vers une mesure désignant l'optimum recherché (au sens de la fonction d'énergie associée à la mesure de Gibbs).

Une analyse fine des vitesses de convergence et des écarts asymptotique permet alors d'aboutir à diverses conditions de convergence, dont celle de Hajek : si le schéma de température utilisé vérifie, alors l'algorithme converge en probabilité vers l'optimum global.

En pratique, ce schéma converge beaucoup trop lentement et l'on préfère des températures à décroissance linéaire ou quadratique, quitte à n'obtenir qu'un minimum local (qu'on espère proche du minimum global).

8-Les champs d'applications

8.1.1 les exemple d'Utilisation du recuit simulé

Comme pour toute métaheuristique, la méthode trouve son application dans de nombreux domaines dans lesquels on a à résoudre des problèmes d'optimisation difficile. On peut citer entre autres la conception des circuits électroniques (placement-routage) ou l'organisation de réseaux informatiques.

En fait, le problème de la restauration d'image est de proposer, à partir de l'image observée, l'image restaurée la plus probable, c'est-à-dire compte tenu de l'expression ci-dessus, celle qui minimise l'énergie $U_y(x)$.

Une technique de recuit simulé s'impose donc. On part d'une configuration initiale qui est celle de l'image observée, sans contours, et l'on modifie les variables pixels et contours par des itérations successives suivant une règle de Monte-Carlo. Les variations d'énergie sont calculées d'après l'expression de $U_y(x)$.

La température décroît régulièrement jusqu'à ce que l'image restaurée soit satisfaisante. Bien entendu, la partie la plus délicate est le choix des différents paramètres définissant la fonction énergie, de même que le plan de décroissance de la température. La figure 9.8, tirée d'un article de S. Geman et D. Geman montre le résultat de la méthode dans un cas très simple.

L'utilisation du recuit simulé est une approche très prometteuse pour le traitement d'image de tomographie ou d'images satellites. Les difficultés actuelles

concernent le choix automatique des nombreux paramètres définissant la fonction énergie, et bien sûr le volume des calculs, que des processeurs parallèles permettraient d'effectuer plus rapidement.

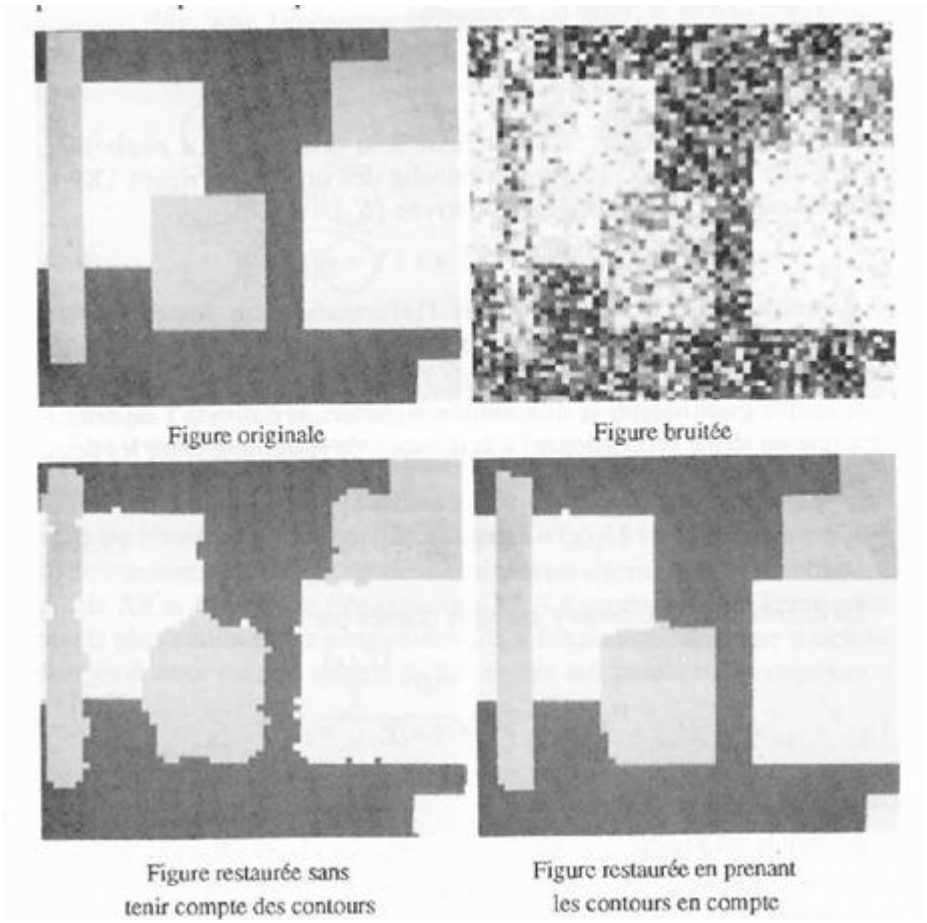


Figure 9.8 Restauration d'image et tracé automatique de contours par recuit simulé (d'après S. Geman et D. Geman, "Bayesian image analysis", in *Disordered Systems and Biological Organization*, E. Bienenstock, F. Fogelman-Soulié et G. Weisbuch eds., *NATO ASI Proceedings*, **F20**, Springer 1986) .

Le lien entre systèmes magnétiques à température finie, automates probabilistes et recuit simulé est particulièrement clair dans les exemples de répartition de fonctions logiques entre circuits intégrés, et de restauration d'image.

Dans ces deux cas, l'énergie apparaît sous la forme d'une somme d'interactions entre les composants, et le processus de mise à jour des variables est celui d'une itération séquentielle aléatoire d'automates probabilistes.

Ce lien est moins clair dans d'autres problèmes d'optimisation combinatoire, comme celui du voyageur de commerce; dans ce cas il se réduit à l'utilisation du recuit simulé. Chacun des trois domaines a bénéficié d'apports des deux autres:

- Les méthodes de calcul de la mécanique statistique, champ moyen et méthode des répliques, viennent des systèmes magnétiques.
- Les méthodes numériques de Monte-Carlo sont en fait des itérations aléatoires d'automates probabilistes.
- Enfin, certains algorithmes classiques de l'optimisation combinatoire ont été utilisés dans la recherche des états d'énergie minimum des verres de spins (algorithme du postier chinois par exemple). **[4]**

8.1.2-Un autre domaine d'application de recuit simulé

Dans le domaine du traitement des images, nous sommes souvent amenés à minimiser une énergie.

La fonction d'énergie présente en général un grand nombre de minima locaux, ce qui complique la tâche de trouver un minimum global. Prenons par exemple la méthode classique de descente de gradient: Elle consiste à, pour chaque itération, faire un pas dans la direction de la plus grande pente vers les basses valeurs de la fonction.

Si, pour un état initial donné, l'algorithme arrive après quelques itérations à un minimum local, il n'y a plus de direction qui fait diminuer l'énergie est l'algorithme se trouve alors piégé.

L'idée de l'algorithme de recuit simulé est de donner la possibilité de sortir des minima locaux en autorisant l'algorithme à prendre des pas dans une mauvaise direction avec une faible probabilité.

8.1.3-Le paramètre de température

Le terme recuit vient de l'opération qui consiste à faire fondre, puis laisser refroidir lentement un métal. Cette procédure amène le métal à son état de plus basse énergie, un état très ordonné, tandis qu'un refroidissement trop rapide pourrait laisser le métal piégé dans un état peu favorable, car trop énergétique.

Le recuit simulé cherche donc à imiter cela en introduisant un paramètre global, T, appelé température. A chaque instant, le prochain pas se fera alors au hasard, selon une distribution de probabilité qui dépend de la température. Nous introduisons la distribution de Gibbs avec paramètre de température, définie de la manière suivante :

$$P_T(X= x) = \frac{1}{Z(T)} e^{\frac{-U(x)}{T}} \text{ , avec } Z(T) = \sum e^{\frac{-U(x)}{T}}$$

La distribution de Gibbs a la bonne propriété de se comporter de la même manière que le modèle physique à températures extrêmes. Pour T très grande, $e^{\frac{-U(x)}{T}}$ se rapproche de 1 et P_T converge vers la probabilité uniforme sur le domaine. Toute direction a donc la même probabilité d'être prise. Pour T proche de 0, on montre que P_T est uniformément distribuée sur les minima globaux. Les deux extrêmes correspondent à un algorithme qui cherche le minimum complètement au hasard et à l'algorithme de descente de gradient respectivement.

L'idée d'introduire un paramètre de température à été proposée par Kirkpatrick .

8.1.4-Echantillonneur de Gibbs

Afin de réaliser l'algorithme du recuit simulé, nous avons besoin d'une méthode d'échantillonner selon la loi de Gibbs.

L'échantillonneur de Gibbs, proposé par Geman , procède en créant itérativement une suite de B-splines (toutes cherchant à modéliser un même contour).

Après un nombre d'itérations suffisamment grand le contour créé est une réalisation de la loi de Gibbs. A l'étape n , l'algorithme consiste à effectuer les opérations suivantes:

- On choisit un point de contrôle du spline,
- Pour ce point, on calcule la probabilité locale conditionnelle,
- Les coordonnées du point sont mis à jour par un tirage aléatoire selon la loi trouvée.

Dans la forme originale de l'échantillonneur de Gibbs, on traite des images considérées comme étant des champs de Markov et la probabilité conditionnelle locale d'un pixel ne dépend alors uniquement de la configuration du voisinage. Ici, le calcul de la probabilité conditionnelle locale est plus compliqué et sera traité ultérieurement.

Si le choix du point de contrôle à l'étape n se fait de manière à ce que tous les points soient balayés un grand nombre de fois, le splines obtenu après une infinité d'itérations est une réalisation de la loi de Gibbs.

La convergence de l'algorithme a été démontré par Winkler (1995). En pratique, nous allons considérer que l'on a convergé lorsque les changements sont petits.

8.2-L'algorithme du recuit simulé

Nous disposons maintenant des outils nécessaires pour présenter l'algorithme du recuit simulé proposé par Geman . Il se présente comme ceci:

°On choisit une température initiale T_0,

°On choisit une B-spline initial,

Pour n = 1

A partir de la configuration précédente x(n-1), on utilise l'échantillonneur de Gibbs pour obtenir une réalisation de la loi de Gibbs d'énergie U(x)/T(n),

°On diminue la température et incrémente n,

°On arrête lorsque les changements sont petits.

La température initiale ne doit pas être choisi trop faible, pour assurer la convergence, mais pas non plus trop élevée, pour assurer un temps de calcul raisonnable. Puisque l'algorithme ne reste pas bloqué dans des minima locaux, la B-spline initiale peut être choisie quelconque.

En général, on fait un balayage de tous les points de contrôle à chaque niveau de température.

Si la température décroît suffisamment lentement, on a convergence de l'algorithme. En théorie, il faut une décroissance logarithmique, mais en pratique on utilise des décroissances géométriques avec des bons résultats.

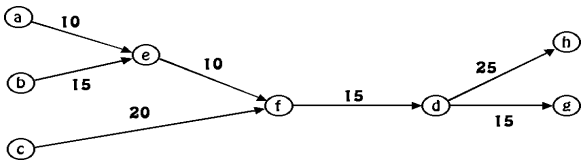
8.3-L'échantillonnage conditionnelle

Dans le cas classique d'un champ de Markov discret, il est facile de calculer l'énergie associée à chaque valeur possible d'un pixel et d'en déduire la probabilité conditionnelle locale.

Or, appliqué à l'estimation des contours par des B-splines, nous avons tellement de valeurs, ou emplacements, possibles pour chaque point de contrôle que l'on peut considérer le problème comme étant continue. Puisque chaque calcul d'énergie coute cher en temps de calcul, nous devons faire un choix rusé d'emplacements pour lesquels les calculs seront faits. Il n'est pas optimal d'utiliser un quadrillage uniforme d'emplacements, car à température basse la distribution de probabilité conditionnelle est très localisée est la majorité des points retenus ne serviront à rien à cause de leur trop grande distance du centre de la distribution. Ce problème a été traité par Jamieson , qui a choisi d'utiliser des méthodes adaptatives pour y remédier. Ces méthodes consiste à prendre les énergies calculées précédemment comme le point de départ, puis d'identifier la région de plus grande incertitude pour y placer d'avantage de points.

8.4-Affectation optimale de tâches sur des processeurs répartis : méthode de recuit simulé

L'affectation optimale de tache sur les processeur d'un système réparti est un problème dont la résolution effective est importante durant toutes les phase du développement de tels système .au cours de la phase de conception, elle permet d'aider à la détermination de la configuration permettant d'atteindre un niveau de performance souhaite ;pendant la phase de fonctionnement ,elle facilite l'optimisation des ressources disponibles ;au cours d'une reconfiguration ,elle aide à la réaffectation des modules en fonction des changements affectant le système Le problème peut être représenté par deux graphes comme on peut le voir sur la figure ci-dessous. Le graphe des taches montre les interdépendances entre taches et les taux des transferts d'informations associés .L'autre graphe correspond au réseau sur lesquels les processeurs sont connectés.



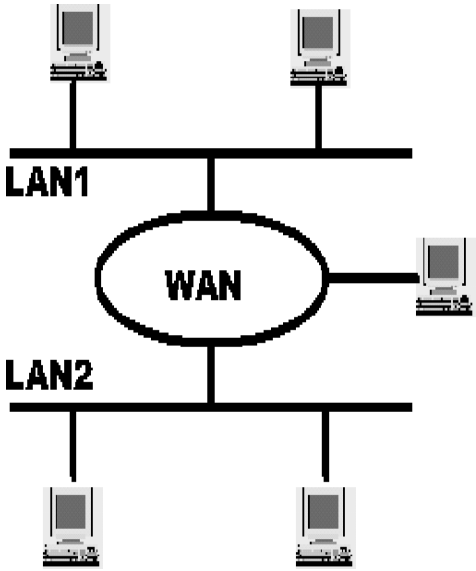


Figure 3 :(a) Un exemple de graphe de communication entre tache.(b) Un exemple de réseau de communication

Multiprocesseur .Les algorithme existants appartiennent aux catégories suivantes :

- algorithme exacts appliqués à des problèmes simplifié

- algorithme heuristique sous-optimaux appliqués aux problèmes complets ou simplifiée

- algorithme exacts appliqués aux problèmes complets.

Cependant, le problème étant NP- complet, sa résolution exacte utilisant par exemple un schéma de type séparation et évaluation n'est pas envisageable pour des problèmes d'une taille réelle , c'est pourquoi l'utilisation du recuit simulé (qui a démontré sa réelle capacité à trouver des solution quasi-optimales dans beaucoup de problèmes d'optimisation combinatoire) est largement justifiée.

Nous avons donc résolu ce problème en utilisant le recuit simulé, méthode dont nous avons une longue expérience. En outre, des approches de type séparation et évaluation appliquées à ce genre de problèmes sont disponibles dans la littérature. Elles sont essentiellement basées sur la résolution d'un problème non linéaire d'évaluation. A l'heure actuelle nous développons notre propre approche de type séparation et évaluation mais à partir d'une reformulation linéaire du problème. **[5]**

Conclusion

Ce mémoire de fin d'étude a eu comme objectif la résolution d'un problème combinatoire en utilisant le récuit simulé . Pour conclure on commencera de donner les parties qui ont été clarifiées dans ce mémoire afin de comprendre le contenu. Le travail présenté s'articule autour de cinq parties :

Dans la première partie, nous avons étudié l'optimisation combinatoire, dans laquelle nous avons vu les problèmes classiques d'optimisation combinatoire , dans la deuxième partie, nous avons présenté les méthodes de résolutions exactes (méthode de Branch and Bound, méthode les coupes planes et les coupes de Gomory) , dans la troisième partie, nous avons expliqué les Métaheuristiques et leurs classifications possibles , dans la quatrième partie, nous avons étudié en détail le recuit simulé et dans la cinquième partie, nous avons éclairci les champs d'applications , en montrant : les exemples d'utilisations du recuit simulé.

Enfin Nous avons essayé de donner une idée générale de recuit simulé et ses utilisations multiples.

Les Références

. [1] : C.H. PAPADIMITRIOU, K. STEIGLITZ, Combinatorial optimization - algorithms and complexity. Prentice Hall, 1982.

. [2] Cours des Méthodes de Résolution Exactes Heuristiques et Métaheuristiques

.[3] coupe_gomory

. [4] L'article original de S. Kirkpatrick, C. Gelatt et M. Vecchi, "Optimization by smulated Annealing",1983

. [5] : Travaux de recherche (1995-1999)

. [5] Personnes impliquées : Yskandar Hamam