République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Akli Mohand Oulhadj - Bouira -

Faculté des Sciences et des Sciences Appliquées
Département de Mathématiques

Mémoire de Master

Filière : Mathématiques

Spécialité : Recherche Opérationnelle

Thème

Les méthodes déterministes et les méthodes stochastiques de l'optimisation globale.

Présenté par :
Djahnit Anis et Hadadi Yacine

Devant le jury d'examen composé de :

M^{me}	**Boudref Mohamed Ahmed**	**Présidente**	**MCA**	**UAMO Bouira**
M^{me}	**Bekri Houria**	**Promotrice**	**MAA**	**UAMO Bouira**
M^{me}	**Alem Lala Maghnia**	**Examinatrice**	**MCB**	**UAMO Bouira**
M^{me}	**Ouiza Imine**	**Examinatrice**	**MAA**	**UAMO Bouira**

2020/2021

Remerciements

La réalisation de ce mémoire à été possible grâce au concours de plusieurs personnes à qui je voudrais témoigner toute ma reconnaissance.

Tout d'abord, j'exprime ma gratitude et mes remerciements à notre promotrice M^{me} Bekri Houria pour ses conseils. Et l'aide de mettre à notre disposition tout ce dont nous avons besoin pour faire ce travail, pour sa patience, sa disponibilité et surtout ses judicieux conseils, qui ont contribué à alimenter ma réflexion.

Je tiens à remercier sincèrement les membres du jury d'avoir fait partis de ces membres. qui me font le grand honneur d'évalue ce travail. En fin, je tiens à témoigner toute ma gratitude à pour leur confiance et leur support estimable.

Dédicaces

Nous tenons à remercier tout premièrement Dieu le tout puissant pour la volonté, la santé et la patience, qu'il nous a donné durant toutes ces longues années.

Ainsi, nous tenons également à exprimer nos vifs remerciements à :

- Mes très chères parents pour toute leurs éducation, leurs sacrifices, leurs amour, leur tendresse, leurs soutient tout au long des mes études et qui m'ont donné un magnifique modèle de labeur et de persévérance.
- Mes chères sœurs , pour leurs encouragements et leur soutien moral.
- Mes chères frères, Abdelhamid, pour leurs appuis et leurs encouragements.
- Toute ma famille pour leurs soutiens tout au long de mon parcours universitaire.
- A mon binôme et mon ami : Yacine.
- Mes amis et à tous ceux qui me connaissent et qui m'ont aidé à réaliser ce travail.

ANIS .

Dédicaces

Je dédie ce modeste travail à :

- Mes très chères parents pour toute leurs éducation, leurs sacrifices, leurs amour, leur tendresse, leurs soutient tout au long de mes études et qui m'ont donné un magnifique modèle de labeur et de persévérance.
- Mes chères sœurs pour leurs encouragements et leur soutien moral.
- Mes chères frères, pour leurs appuis et leurs encouragements.
- Toute ma famille pour leurs soutiens tout au long de mon parcours universitaire.
- Mes amis et à tous ceux qui me connaissent et qui m'ont aidée à réaliser ce travail.

Yacine .

TABLE DES MATIÈRES

INTRODUCTION GÉNÉRALE

Durant les 20 dernières années, le domaine de la recherche sur "l'optimisation global" s'est considérablement enrichi grâce, notamment, a l'accroissement de la puissance de calcul des ordinateurs. Ces progrès ont permis de résoudre des problèmes auparavant insolubles.

L'optimisation globale recherche la meilleure solution du domaine en entier, c'est-a-dire que dans tout le domaine il n'existe aucune solution qui lui soit meilleure tout en respectant les contraintes. Cette solution est appelée l'optimum global.

Il n'y a pas si longtemps, mettre au point des méthodes numériques permettant de déterminer la solution d'un problème d'optimisation non linéaire et non convexe pouvant répondre à un ensemble de contraintes, elles aussi non linéaires et non convexes, paraissait très difficile. Bien que la théorie classique de l'optimisation ne peut pas être appliquée directement dans les problèmes d'optimisation globale, les outils traditionnels tels que l'analyse convexe, sont largement utilisés dans la construction des méthodes d'optimisation globale. Cette approche constitue une partie importante de l'optimisation globale déterministe. Par exemple, un remarquable progrès a été accompli dans la construction des algorithmes de minimisation des fonctions concaves dans des régions convexes.

Ces dernières années, de nombreux travaux ont été réalisés. On peut les classer en deux grandes familles :

Les Méthodes déterministes, comme leur nom l'indique, nous offrent la certitude d'obtenir l'optimum global et ne laissent aucune place au hasard et conduiront pour un contexte initial donné à une même solution finale. Pour ces méthodes, l'exploration de l'espace de solutions se fait grâce à des procédures de recherche qui sont élaborées à partir de la constante de Lipschitz, des dérivées ou d'autres informations locales et globales concernant la fonction objectif. Les modèles déterministes ne traitent pas adéquatement les informations disponibles sur la fonction objectif.

Les Méthodes stochastiques peuvent souvent faire face à ce genre de problèmes plus facilement et plus efficacement que les algorithmes déterministes. Les algorithmes stochastiques explorent l'espace de solutions grâce en partie à des procédures de transitions aléatoires. Ainsi, plusieurs exécutions successives de ces algorithmes, pourront conduire à des résultats différents (pour un même point initial). L'avantage des méthodes stochastiques est leur simplicité et leur

pertinence pour les problèmes où les évaluations de la fonction objectif sont corrompues par un bruit aléatoires, ainsi que leur robustesse à l'égard de la croissance de la dimension. L'inconvénient majeur de ces méthodes est qu'elles peuvent diverger, et passer plusieurs fois à côté de la solution. L'obtention de l'optimum global n'est pas garantie, il est seulement repéré avec une probabilité proche de 1. De nombreux algorithmes, où l'aléatoire et l'argument statistique sont impliqués ont été proposés. Ces algorithmes sont basés sur des analogies avec les processus naturels. Les exemples les plus connus de tels algorithmes sont L'algorithme génétique, le recuit simulé, et la recherche tabou. Les algorithmes stochastiques d'optimisation globale sont très populaires dans les applications.

Dans ce mémoire, on s'intéresse à quelques méthodes d'optimisation globale déterministes et non déterministes ou stochastiques. Nous nous sommes concentrés sur la résolution de problèmes explicites, c'est-à-dire des problèmes dans lesquels toutes les expressions des équations de la fonction objectif sont connues de façon explicite.

Dans le premier chapitre de notre étude nous présenterons quelques méthodes déterministes d'optimisation globale basées sur l'utilisation des techniques de partition et d'élimination (Branch-and-Bound). Notre attention sera portée sur les méthodes de recouvrement ou lipschitziennes qui ont la réputation d'être efficaces en dimension 1. Parmi ces méthodes, il y a celles qui sont basées sur la construction des fonctions sous-estimateurs de la fonction objectif sur le domaine faisable (Piyavskii). Aussi, une méthode de recouvrement itérative d'Evtushenko appliquée à la classe de fonctions lipschitziennes.

Dans le second chapitre nous étudierons les différentes méthodes stochastiques, comme l'algorithme génétique et le recuit simulé .

En fin, nous terminerons notre travail par une conclusion générale sur les différentes méthodes utilisées en optimisation globale.

CHAPITRE 1

LES MÉTHODES DÉTERMINISTES D'OPTIMISATION GLOBALE

1.1 Introduction

Dans ce chapitre nous allons présenter trois méthodes d'optimisation globale déterministes, en donnant leurs principes ainsi que leurs algorithmes de bases. La première est de type Séparation-Évaluation, connue sous sa dénomination anglaise Branch and Bound. Les deux autres sont les méthodes les plus connues de recouvrement qui s'appliques aux fonctions lipschitziennes : Piyavskii et Evtushenko.

1.2 La méthode Branch and Bound

1.2.1 Historique

Les algorithmes branche and bound ont pris une place non négligeable dans le monde de l'optimisation globale. Les raisons de cet engouement sont simples, c'est un principe a la fois basique et général sur lequel peut venir se greffer de nombreuses idées pour améliorer la convergence. Dans notre étude nous intéresserons qu'a la méthode branch and bound . B&B est une méthode pour résoudre une classe de problèmes d'optimisation globale, il s'agit de la recherche par décomposition du domaine en sous-domaine. Cette technique repose sur l'utilisation des intervalles puisqu'à chaque étape, le problème courant est traité de manière globale, c'est à dire que le domaine (intervalle ou pavé) est considérer comme élément du calcul pour l'évaluation.

1.2.2 Introduction

Branch and Bound est un algorithme assez général qui joue un rôle très important dans la théorie d'optimisation globale. L'idée générale de la méthode est de décomposer le problème primaire en sous problèmes parallèles (Branching) qui peut être graduellement plus facile à résoudre, puis évaluer les bornes inférieures et supérieures (Bounding) des valeurs des solutions optimales sur ces problèmes secondaires. Par conséquent, le procédé Branch and Bound peut

être représenté sous forme d'un arbre : le problème initial est situé comme racine de cet arbre et les branches sont les sous problèmes hiérarchiquement construits par l'algorithme. Cette construction est régie par la stratégie de la recherche qui déterminera la suite de solutions des problèmes secondaires. La séquence de la décomposition et la recherche de la solution continue jusqu'à ce qu'il puisse vérifier que l'un ou l'autre sur la branche indiquée ne peut pas apporter une meilleure solution que celle du candidat sortant (trouvé déjà par le procédé de Branch and Bound). Notons que Branch and Bound a été à l'origine développée pour résoudre des problèmes de la programmation entière. Plus tard, cet algorithme a été appliquée avec succès dans des problèmes très difficiles en optimisation globale .

1.2.3 Le principe de la méthode branch and bound

Soit (P) le problème d'optimisation globale :

$$(P)\begin{cases} \min f(x) \\ x \in \mathrm{H} \end{cases}$$

Où : H est un compact de $\mathbb{R}^n$.
$f : \mathrm{K} \mapsto \mathbb{R}$ $(\mathrm{H} \subseteq \mathrm{K} \subseteq \mathbb{R}^n)$ f continue et non convexe.

L'algorithme de B&B consiste à engendrer deux suites convergentes $\{UB_k\}$ et $\{LB_k\}$ des bornes supérieure et inférieure respectivement de la valeur minimale de la fonction du problème (P).

$$\begin{cases} UB : Upper\ bound \\ LB : Lower\ bound \end{cases}$$

Une relaxation initiale R de l'ensemble réalisable H sera définie telle que : $\mathrm{H} \subset \mathrm{R}$.
R est convexe, il peut être un simplexe, un rectangle, un cône, ...etc. A chaque itération k les problèmes des bornes inférieure et supérieure seront résolus sur un nombre fini de sous-ensembles de R. On notera ces sous-ensembles $\mathrm{R}_{k_i} \in \mathbf{I}_k$ où I_k est l'ensemble des sous-ensembles actifs à l'itération k. Sur chaque sous-ensemble R_{k_i} les bornes inférieure et supérieure LB_k et UB_k seront calculées par la relaxation de f sur R_{k_i} et la relaxation de $\min f$ localement sur le sous ensemble réalisable $\mathrm{R}_{k_i} \cap \mathrm{H}$ respectivement. Cette méthode utilise la stratégie "le meilleur d'abord".
En effet, les bornes inférieure et supérieure finales pour l'itération k seront données par :

$$\begin{cases} LB_k = \min LB_{k_i} \\ UB_k = \min UB_{k_i} \end{cases}$$

respectivement, et tout sous-ensemble sur lequel la borne inférieure dépasse UB_k sera éliminé, car $\min f$ ne peut être atteint sur un tel sous-ensemble.

Au fait, cette méthode peut se représenter schématiquement par une arborescence qui a pour racine l'ensemble R, et pour sommets les sous-ensembles R_{k_i} qui s'obtiennent par les subdivisions successives, et deux sommets seront reliés si et seulement si le deuxième sous-ensemble est obtenu par la partition directe du premier, et à chaque niveau de l'arborescence créé les bornes inférieure et supérieure seront obtenues par l'application d'une recherche locale.

Notons $x^\star$ la solution optimale du problème (P) pour ce qui suit.

1.2.4 L'algorithme de base de la méthode branch and bound

On peut résumer la procédure précédente par les étapes suivantes :

Algorithme général

i. Construire l'ensemble R tel que : H $\subset$ R.

ii. Posons : k=1, $\mathrm{I}_k = \mathrm{R}$, fixer $\varepsilon > 0$.

iii. Construire les problèmes des bornes inférieure et supérieure de $\min f(x)$ sur R. Soient LB_k, UB_k les solutions obtenues respectivement.

iv. Si : $UB_k - LB_k \leq \varepsilon$, donc on s'arrête et on pose :
$\min f(x) = UB_k$ et $x^\star = x^k \in \{x : f(x) = UB_k, \quad x \in \mathrm{H} \cap \mathrm{R}\}$

v. Sinon, subdiviser I_k en deux sous-ensembles (ou en un nombre fini de sous-ensembles) R_{k_1} et R_{k_2} tels que :

$$\bigcup_{i=1}^{i=2} \mathrm{R}_{k_i} = \mathrm{R} \quad et \quad \dot{\mathrm{R}}_{k_1} \cap \dot{\mathrm{R}}_{k_2} = \emptyset$$

où $\dot{\mathrm{R}}$ est l'intérieur de R.

vi. Construire les problèmes des bornes inférieure et supérieure de $\min f(x)$ sur $\mathrm{H} \cap \mathrm{R}_{k_i}$, $i = 1, 2$. Soient LB_{k_1}, UB_{k_1} et LB_{k_2}, UB_{k_2} les solutions obtenues.

vii. Posons :

$$\begin{cases} UB_{k+1} = \min \{UB_{k_1}, UB_{k_2}, UB_k\} \\ LB_{k+1} = \min \{LB_{k_1}, LB_{k_2}\} = LB_{k^\star} \end{cases}$$

viii. Posons : $\mathrm{I}_k = \{\mathrm{R}_{k_1}, \mathrm{R}_{k_2}\}$.

ix. Éliminer de I_k tout sous-ensemble $\mathrm{R}_{k_j}, \quad j = 1, 2$, tel que :

$$LB_{k_j} > UB_{k+1} \quad où \quad \mathrm{H} \cap \mathrm{R}_{k_j} = \emptyset$$

et posons : $\mathrm{I}_{k+1} = \mathrm{R}_{k_{i^\star}}$.

x. Posons : $k = k + 1$ et revenant à la quatrième étape de l'algorithme.

Notation :

R_k : Le sous-ensemble actuel.
LB_k : La borne inférieure (à la k^{ieme} itération).
UB_k : La borne supérieure (à la k^{ieme} itération).
x^k : La solution trouvée (à la k^{ieme} itération).

1.2.5 La convergence de la méthode branch and bound

Évidemment si l'algorithme précédent se termine à l'itération j, alors x^j est la solution optimale et UB_j est la valeur optimale de la fonction objectif, mais en général on ne peut pas garantir ça, c'est à dire le fait de s'arrêter après un nombre fini d'itérations, et si l'algorithme est infini, alors il engendre au moins une suite $\{\mathrm{R}_k\}$ infinie des sous-ensembles des subdivisions successives telle que : $\mathrm{R}_{k+1} \subset \mathrm{R}_k$, $k \in \mathbb{N}$.

Donc on doit montrer que chaque point d'accumulation de la suite des solutions $\{x^k\}$ correspondante est une solution optimale du problème donné.

Le théorème suivant démontre la convergence de l'algorithme de branch and bound.

Théorème 1.1. *Si pour chaque suite infinie $\{R_k\}$, $\mathrm{R}_{k+1} \subset \mathrm{R}_k$, $k \in \mathbb{N}$ des ensembles des partitions successives, les bornes inférieure et supérieure vérifient :*

$$\lim_{k\to\infty}(UB_k - LB_k) = \lim_{k\to\infty}(UB_k - LB(\mathrm{R}_k)) = 0, \tag{1.1}$$

alors

$$UB = \lim_{k\to\infty} UB_k = \lim_{k\to\infty} f(X_k) = \lim_{k\to\infty} LB_k = LB. \tag{1.2}$$

Et chaque point d'accumulation X^* de la suite $\{X_k\}$ est une solution optimale de $\min f(X)$, $X \in \mathrm{H}$.

Preuve. A l'itération k le sous-ensemble R_k sera choisi à partir de la règle suivante :

$$LB_{k+1} = \min\{LB_{k_1}, LB_{k_2}\}$$

de l'algorithme ci-dessus, à la fin de l'itération (k-1) et donc :

$$LB_k = LB(\mathrm{R}_k).$$

Soit $\{X_k\}$ la suite des solutions optimales engendrées par l'algorithme, comme H est compact, alors $\{X_k\}$ a des points d'accumulations.
Soit X^* un point d'accumulation de la suite $\{X_k\}$, donc il existe une sous-suite infinie de $\{X_k\}$ qui converge vers X^*, et comme f est continue alors :

$$\lim_{k\to\infty} f(X_k) = f(X^*).$$

Posons : $f^* = \min\{f(X), X \in \mathrm{H}\}$, la suite $\{LB_k\}$ des bornes inférieures est croissante monotone, majorée par f^*, donc la suite $LB = \lim_{k\to\infty} LB_k$ existe.
D'autre part, la suite $\{UB_k\}$ des bornes supérieures est décroissante, minorée par f^*, donc sa limite : $UB = \lim_{k\to\infty} UB_k$ existe, et on a : $UB_k = f(X_k) \geq f^*$, et ça implique que :

$$LB \leq f^* \leq \lim_{k\to\infty} f(X_k) = f(X^*) = UB.$$

Et du fait de (1.1) on peut déduire directement (1.2).
Chaque réalisation de l'algorithme de Branch and Bound doit donc spécifier :

i. L'ensemble R tel que : H $\subset$ R.

ii. Les procédures qui donneront les bornes inférieure et supérieure sur les sous-ensembles engendrés par l'algorithme.

iii. Les subdivisions successives de R en sous-ensembles.

Bien entendu, ça va dépendre de la structure du problème (P). Pour cela nous allons étudier chaque cas à part, en se basant sur les trois points précédent, et en montrant la convergence de l'algorithme à chaque fois.

1.2.6 Exemples d'applications :

Quelques exemples de problèmes à une seule variable dans l'intervalle [a,b], avec a,b dans R pour certaines fonctions lipschitziennes étudiés, trouvés dans la littérature précisément dans l'article [14], et leurs résultats après l'application de l'algorithme de branch-and-bound unidimensionnel .

Nombre de la fonction	La fonction de Lipschitz f(x)	l'intervalle [a,b]	source
1.1	$x^6 - 15x^4 + 27x^2 + 250$	[-4,4]	(14)
1.2	$-(-3x + 1.4)sin(18x)$	[0,1]	(14)

Table 1 : Test des fonction de Lipschitz a une seule variable.

Nombre de la fonction	valeure optimale f^{opt}	minimum globale x^{opt}	itérations iter
1.1	7	3	12
1.2	-1.4891	0.9661	

Table 2 : L'optimum et la valeur optimale du Test des fonction de Lipschitz a une seule variable

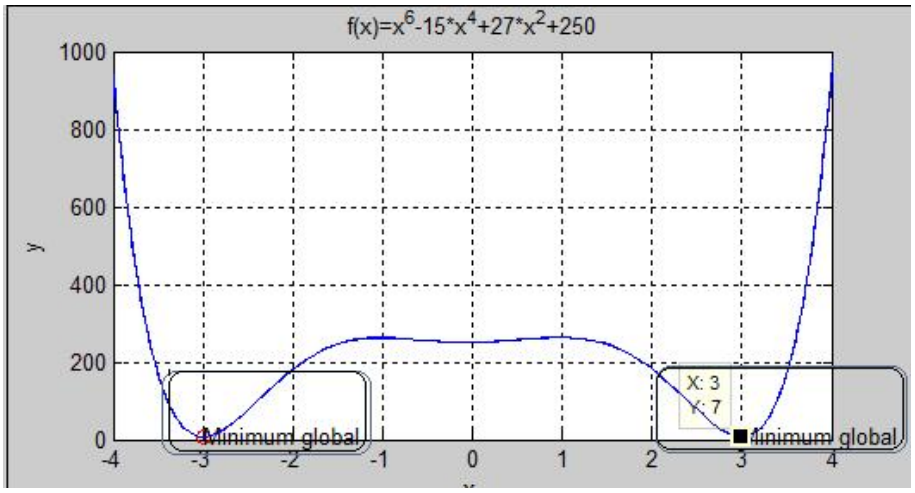

FIGURE 1.1 – Les minimums globaux de la foncion f (1.1)

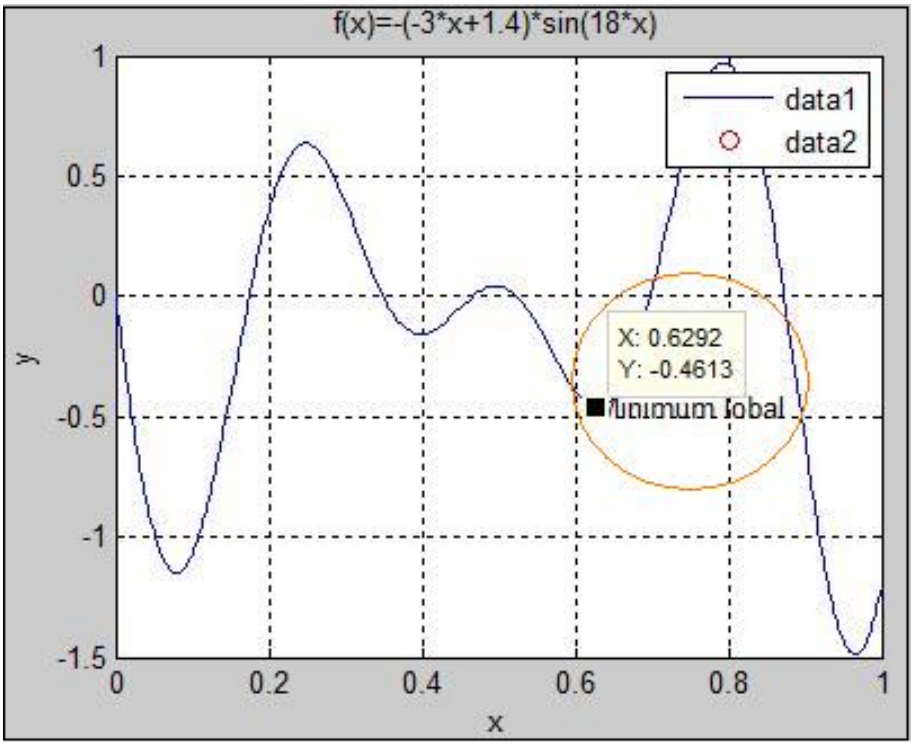

FIGURE 1.2 – Les minimums globaux de la foncion f (1.2)

1.3 Les méthodes lipschitziennes

1.3.1 Introduction

Soit (X,d) un espace métrique et f une fonction lipschitzienne de constante L sur X. La propriété suivante est donc vérifiée :

$$\forall x, y \in X \quad |f(x) - f(y)| \leq Ld(x, y)$$

Si nous arrivons à choisir des points $x_1, x_2, \ldots, x_k$. On pose $m_k^* = \min\{f(x_1), f(x_2), \ldots, f(x_k)\}$ et on désigne par $(B(x_i, \rho_{i,k}))_{0 \leq i \leq k}$ la famille des boules de centres x_i et de rayons $\rho_{i,k} = \frac{f(x_i) - m_k^* + \varepsilon}{L}$ pour $0 \leq i \leq k$ avec ε un nombre positif donné. Alors pour tout $i \in \{1, 2, \ldots, k\}$ et pour tout $x \in B(x_i, \rho_{i,k})$ on a :

$$d(x, x_i) \leq \frac{f(x_i) - m_k^* + \varepsilon}{L}$$

alors

$$Ld(x, x_i) \leq (f(x_i) - m_k^* + \varepsilon)$$

donc

$$f(x_i) - f(x) \leq (f(x_i) - m_k^* + \varepsilon)$$

ce qui équivalent à

$$m_k^* - \varepsilon \leq f(x)$$

Il en résulte que si $X \subset \bigcup_{i=1}^k B(x_i, \rho_{i,k}), m_k^*$est un minimum global de f avec la précision ε.

Théorème 1.2. *Soit f une fonction lipschitzienne de constante L, definie sur un compact X de l'espace $(\mathbb{R}^n, d)$. Supposons que la fonction f soit évaluée aux points $x_1, x_2, \ldots, x_k \in X$ et on pose $m_k^* = \min\{f(x_1), f(x_2), \ldots, f(x_k)\}$. Si on a $X \subset \bigcup_{i=1}^k B(x_i, \rho_{i,k})$ alors m_k^* est un minimum globol de f avec la précision ε.*

1.3.2 Méthode de Piyavskii

Introduction

Considérant (P) le problème de minimisation globale :

$$(P) \begin{cases} \min f(x) \\ x \in \mathrm{X} \end{cases}$$

Où : X est un compact de $\mathbb{R}^n$, et $f : \mathrm{X} \mapsto \mathbb{R}$ une fonction continue et non convexe.
Dans cette section, on considère le problème de minimiser une fonction f lipschitzienne sur un compact X, c.à.d une fonction qui satisfait la condition de Lipschitz. Des papiers et livres

proposent plusieurs approches pour la résolution numérique de ce problème et donnent une classification des problèmes et de leurs méthodes de résolution. Dans cette section, on va présenter l'algorithme de Piyavskii qui est un algorithme pour optimiser une fonction d'une seule variable sur un intervalle [a ,b] . [1]

Définition 1.1. Une fonction $f : P \subset \mathbb{R}^n \longrightarrow \mathbb{R}$ est dite lipschitzienne sur P, s'il existe une constante réelle $L = L(f, P)$ tell que :

$$\forall x, y \in P, \quad f(x) - f(y) \leq L \|x - y\| \tag{1.3}$$

où L est une constante appelée la constante de Lipschitz

Définition 1.2. On dit qu'une fonction g est un sous-estimateur de la fonction f sur P si :

$$\forall x \in P, \quad g(x) \leq f(x)$$

D'après l'inéquation (1.3) on a :

$$f(y) - L \parallel x - y \parallel \leq f(x) \tag{1.4}$$

Fixons $y \in P$, il résulte de l'inéquation (1.4) que la fonction concave : $F(x) = f(y) - L \parallel x - y \parallel$ est un sous-estimateur de f sur P. Nous allons construire une suite de points $\{x_k\}$ et une famille d'ensembles $\{E_k\}$ vérifiant $P \subset \cup_k E_k$ de la manière suivante :
Considérons la fonction

$$F_1(x) = f(x_1) - L \parallel x - x_1 \parallel \tag{1.5}$$

Où x_1 est un point quelconque de P. Le point suivant de l'itération est défini par :

$$x_2 = \arg\min_{x \in P} f(x)$$

En remplaçant x_1 par x_2 dans l'inéquation (1.5), et on obtient un autre sous-estimateur de $f(x)$, qui est :

$$F_2(x) = f(x_2) - L \parallel x - x_2 \parallel \tag{1.6}$$

mais encore un nouveau sous-estimateur qui est donné par :

$$\max_{1 \leq i \leq 2} (f(x_i) - L \parallel x - x_i \parallel)$$

et il est plus proche de $f(x)$ voir la Figure 1.3.
On prend donc

$$F_2(x) = \max_{1 \leq i \leq 2} (f(x_i) - L \|x - x_i\|) \quad et \quad x_3 = \arg\min_{x \in P} F_2(x)$$

A l'étape k, on prend

$$F_k(x) = \max_{1 \leq i \leq k} (f(x_i) - L \|x - x_i\|) \tag{1.7}$$

et

$$x_{k+1} = \arg\min_{x \in P} F_k(x) \tag{1.8}$$

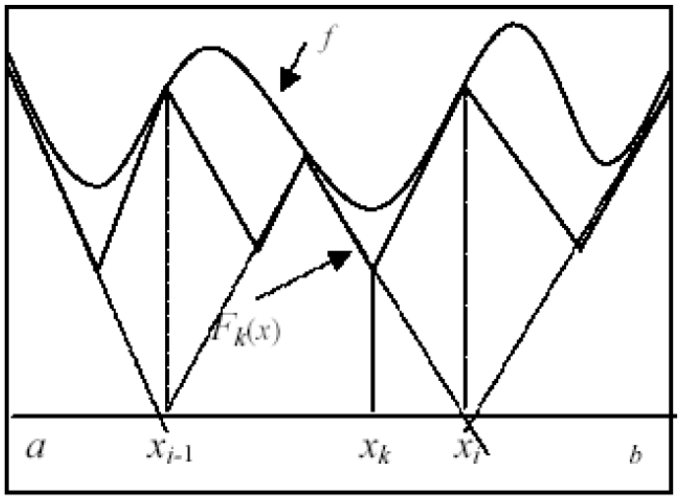

FIGURE 1.3 – Construction des fonctions borne

Dans la méthode de Piyavskii, on prend comme recouvrement de P la famille d'ensemble suivant

$$\begin{aligned} E_k &= \left\{ x \in \mathbb{R}^n : \max_{1 \le i \le k} \left(f\left(x_i\right) - L \left\| x - x_i \right\| \right) \ge f_k^* - \varepsilon \right\} \\ &= \left\{ x \in \mathbb{R}^n : F_k(x) \ge f_k^* - \varepsilon \right\} \end{aligned} \tag{1.9}$$

Où f_k^* est une approximation du minimum globale de f. Dans le cas unidimensionnel, la méthode de Piyavskii-Shubert consiste a construire des recouvrement de P=[a,b] à partir des fonctions $F_k(x)$. Elle utilise la suite de point $\{x_k\}$ définie par le terme général

$$x_{k+1} = \arg\min_{x \in [a,b]} F_k(x)$$

avec

$$x_1 = \frac{a+b}{2}$$

et

$$F_k(x) = \max_{1 \le i \le k} \left(f\left(x_i\right) - L \left| x - x_i \right| \right)$$

dans ce cas $F_k(x)$ est un sous-estimateur linéaire par morceaux.

Algorithme de Piyavskii

i. Initialisation
poser k=1, $x_1 = \dfrac{(a+b)}{2}$, $x_e = x_1$, $f_e = f(x_e)$.
$F_e = f_e - \dfrac{L}{2}(b-a)$, $F_1(x) = f(x_1) - L\left|x - x_1\right|$ ..

ii. Étape k=2,3....
Si $f_e - F_e \le \varepsilon$ alors arrêter.

Sinon, déterminer $x_{k+1} = arg\min F_k([a,b])$.
Si $f(x_{k+1}) < f_e$, alors poser $f_e = f(x_{k+1})$, $x_e = x_{k+1}$.
poser

$$F_{k+1}(x) = \max_{1\leq i\leq k+1} \left(f\left(x_i\right) - L\left|x - x_i\right|\right) \text{ et } F_e = \min F_{k+1}([a,b])$$

poser k=k+1 et aller à ii

1.3.3 Méthode d'Evtushenko

Evtushenko (1971-1985) a développé et proposé plusieurs méthodes de recouvrement, celle que nous allons présenter est la plus connue et la plus commentée dans la littérature. Elle utilise comme recouvrement des cubes définis par :

$$S_{jk} = \left\{x \in \mathbb{R}^n / d_{\max}\left(x_j, x\right) = \left\|x - x_j\right\|_\infty \leq r_{j,k}\right\}$$

avec

$$r_{j,k} = \frac{f\left(x_j\right) - R_k + \varepsilon}{L\sqrt{n}}$$

où R_k est le record d'ordre k définie par : $R_k = min\{f(x_1), f(x_2), ..., f(x_k)\}$:
Alors, pour tout j = 1, 2, ..., k et pour $x \in S_{jk}$, on a :

$$f(x) \geq R_k - \varepsilon$$

En effet, soit $x \in S_{jk}$, puisque la fonction f est lipschitzienne donc

$$f\left(x_j\right) - f(x) \leq L\sqrt{n} d_{\max}\left(x_j, x\right)$$

Il en résulte que :

$$\begin{aligned} f\left(x_j\right) - f(x) &\leq L\sqrt{n}\frac{f\left(x_j\right) - f_k^* + \varepsilon}{L\sqrt{n}} \\ &\leq f\left(x_j\right) - f_k^+ + \varepsilon \end{aligned}$$

D'où

$$f\left(x_j\right) - f(x) \leq L\sqrt{n} d_{\max}\left(x_j, x\right)$$

Par conséquent

$$\bigcup_{j=1}^{k} S_{j,k} \subset S_k$$

avec $S_k = \{x \in X / f(x) \geq R_k - \varepsilon\}$
Si la réunion des S_{jk} ne couvre pas X, le minimum global est atteint dans

$$X \backslash \bigcup_{j=1}^{n} S_{j,k}$$

Par conséquent, les cubes $S_{J,k}$ peuvent être exclus de l'ensemble faisable X, on cherche la solution dans la partie restante. Le problème est résolu lorsque la réunion des cubes couvre complètement X.

a) Méthode d'Evtushenko dans le cas unidimensionnel :

Dans le cas unidimensionnel ($X = [a; b]$) et pour une fonction lipschitzienne f de constante L, et d'aprés le théorème (1.2) il faut le couvrir avec des intervalles de rayons

$$\rho_{i,k} = \frac{f(z_i) - R_k + \varepsilon}{L}$$

On va définir les points

$$\{x_i\}_{1 \leq i \leq k} \subset X$$

qui sont les centres des intervalles

$$\{I_i\}_{1 \leq i \leq k}$$

tel que $X \subset I_i$

Il est clair qu'il faut prendre

$$x_1 = a + \frac{\varepsilon}{L}$$

parce que à l'initialisation

$$f(x_1) = m_1^*$$

Pour définir x_{i+1}, il faut qu'il ne reste pas des points x tels que :

$$x_i \leq x \leq x_{i+1} \quad x \notin I_i \text{ et } x \notin I_{i+1} \tag{1.10}$$

On a $\rho_{i,k} = \frac{f(x_i) - m_i^* + \varepsilon}{L}$, donc il est clair que nous prenons $x_{i+1} = x_i + \rho_{i,k} + \rho_{i+1,k}$ mais le rayon $\rho_{i+1,k} = \frac{f(x_{i+1}) - m_i^+ + \varepsilon}{L}$ dépend de x_{i+1} qui est inconnu. Cependant, on sait que $\rho_{i+1,k} \geq \frac{\varepsilon}{L}$ donc l'équation (1.10) n'est pas vérifiée, on prend alors :

$$\begin{aligned} x_{i+1} &= x_i + \rho_{i,k} + \frac{\varepsilon}{L} \\ &= x_i + \frac{f(x_i) - m_k^* + \varepsilon}{L} + \frac{\varepsilon}{L}. \end{aligned}$$

On s'arrête quand k vérifié $X \subset \bigcup_{i=1}^{k} I_i$, donc si

$$x_k \leq b \text{ et } \quad x_k + \rho_{k,k} > b$$
$$\Leftrightarrow x_{k+1} - \frac{\varepsilon}{L} > b \Leftrightarrow x_{k+1} > \frac{\varepsilon}{L} + b$$

Parce que dans ce cas $b \in I_k$ et il n'existe pas des points tels que $x_k \leq x \leq b$ et $x \notin I_k$ on a :

$$X \subset \bigcup_{i=1}^{k} I_i$$

Algorithme d'Evtushenko dans le cas unidimensionnel :

1) Initialisation :
Poser $x_1 = a + \frac{\varepsilon}{L}, x_e = x_1, f_\varepsilon = f(x_\varepsilon)$

2) Etape $k = 2, 3, \ldots$
Poser $x_{k+1} = x_k + \frac{2\varepsilon + f(x_k) - f_e}{L}$
Si $x_k > b$ alors, arrêter
Sinon calculer $f(x_{k+1})$
Si $f(x_{k+1}) < f_z$ alors,
Poser $f_z = f(x_{k+1})$ et $x_\varepsilon = x_{k+1}$
Poser $k = k + 1$
Aller à l'étape 2.

b) Méthode d'Evtushenko dans le cas multidimensionnel :

Considérons le problème suivant :

$$f^* = \min_{x \neq P} f(x) \tag{1.11}$$

avec $P = \{x \in \mathbb{R}^n / a \leq x \leq b\}$. La fonction f(x) satisfait la condition de Lipschitz avec la norme euclidienne. L'ensemble des solutions approximatives du problème (1.11) est donné par X_ε. Le cube inscrit dans la sphère S_{jk} est définie comme suit :

$$V_{jk} = \{x \in \mathbb{R}^n / x_j - er_{jk} \leq x \leq x_j + er_{jk}\}$$

où

$$r_{j,k} = \frac{f(x_j) - R_k + \varepsilon}{L\sqrt{n}} \tag{1.12}$$

et

$$e = (1, 1, \ldots, 1)$$

D'après les arguments de la section précédente, si $f(x_j)$ et R_k sont connus, alors on peut exclure le cube V_{jj} du pavé P. S'il arrive que $R_k < R_j$ pour un $k > j$, alors on peut omettre le cube V_{jk} le plus grand qui contient le cube V_{jj}. A partir de l'équation (1.12) il n'est pas difficile de déduire une formule pour calculer le coté du nouveau cube élargi :

$$r_{j,h} = r_{jj} + \frac{R_j - R_k}{L\sqrt{n}}$$

Le problème (1.11) sera résolu lorsque la suite des cubes V_{jk} couvre complètement le pavé P.

L'algorithme d'Evtushenko est très simple et facile, elle ne demande pas un grand espace mémoire, ni des calculs auxiliaires importants dans le cas unidimensionnelle.

Mais dans le cas multidimensionnel l'étude numérique a montré qu'elle est très encombrante, compliquée et pour une dimension $n > 3$, elle nécessite un nombre d'évaluation très élevé ; chose qui fait perdre l'algorithme de son efficacité.

1.3.4 Conclusion

Les méthodes de recouvrements ou lipschitziennes proposées sont simples, plus développées et mieux étudiées que les méthodes d'optimisation globale basées sur une stratégie locale. Leur efficacité a été prouvée dans le cas unidimensionnel mais dans le cas multidimensionnel, elles sont encombrantes et trop compliquées pour la réalisation pratique. En plus, le temps de calcul dépend considérablement du choix de la constante de Lipschitz et des calculs auxiliaires.

CHAPITRE 2

QUELQUES MÉTHODES NON DÉTERMINISTES OU STOCHASTIQUES

Dans ce chapitre, nous exposons quelques méthodes stochastiques dans un cadre général des méta-heuristiques, qui sont apparues au début des années 1980 pour résoudre des problèmes d'optimisation difficiles. Ces méthodes ont en commun certaines caractéristiques :

- Leur nature stochastique leur permet d'explorer plus facilement un espace des solutions de très grande dimension (n > 100)
- Si la fonction est connue analytiquement, le calcul du gradient de la fonction objectif n'est pas nécessaire à ces méthodes ;
- Elles sont inspirées par des analogies avec la nature ;
- Un réglage systématique de leurs paramètres est souvent nécessaire pour assurer la convergence de l'algorithme ;
- Elles sont coûteuses en temps de calcul.
- Les résultats de convergence ne s'appliquent pas en pratique.

Le principe de base d'une méta-heuristique est qu'elle commence par générer aléatoirement une solution dans l'espace de recherche, puis, elle évalue sa valeur par la fonction objectif. Ensuite, elle recommence et compare les deux résultats. Si le nouveau résultat est meilleur, elle le garde. Ainsi de suite jusqu'à ce qu'un critère d'arrêt soit vérifié.

Utilisée telle quelle, cette méthode n'est pas très performante, elle fait uniquement une exploration de l'espace de recherche, sans essayer d'améliorer la qualité des solutions trouvées, ni d'utiliser les informations de ces solutions. Un autre algorithme qui en découle est la marche aléatoire, qui cherche une nouvelle solution dans le voisinage de la première, plutôt que dans l'espace de recherche entier. Au contraire de la recherche stochastique, cette dernière est donc une méthode de recherche locale.

2.1 Les algorithmes génétiques

Les algorithmes génétiques (AG) sont des algorithmes de recherche inspirés des mécanismes de l'évolution naturelle des êtres vivants et de la génétique. Les premiers travaux ont été menés par Holland en 1975 [2] dans l'ouvrage "Adaptation of Natural and Artificial System " qui formalise les algorithmes génétiques dans le cadre de l'optimisation mathématique. La parution de l'ouvrage de David Goldberg en 1989 [3] décrivant l'utilisation de ces algorithmes pour la résolution des problèmes concrets, a permis de mieux faire connaître ces derniers dans la communauté scientifique et a marqué le début d'un nouvel intérêt pour cette technique d'optimisation. Ces méthodes s'attachent à simuler le processus de sélection naturelle dans un environnement défavorable en s'inspirant de la théorie de l'évolution proposée par Darwin en 1859 " Journal of Research in to the Geology and Natural History". Selon ces concepts, lorsqu'une population est soumise aux contraintes d'un milieu naturel, seuls les individus les mieux adaptés survivent et génèrent une descendance. Au cours des générations, la sélection naturelle permet l'apparition d'individus de mieux en mieux adaptés au milieu naturel, au détriment de ceux se montrant notoirement inadaptés, assurant ainsi la pérennité de l'espèce. Cette particularité de l'évolution naturelle : la capacité d'une population à explorer son environnement en parallèle et à recombiner les meilleurs individus entre eux, est empruntée par les algorithmes génétiques.

2.1.1 Principes : Définition et vocabulaire

Par analogie à la génétique, ces algorithmes cherchent à partir d'une population initiale les meilleurs individus. Pour constituer les parents appropriés donnant naissance à des meilleures descendances Enfants, parmi les quelle seront tirée les solutions acceptables du problème traité. Chaque problème d'optimisation est caractérisé par des variables de décision qui conditionnent les décisions à prendre, un objectif à satisfaire et des contraintes à respecter. Contrairement aux méthodes d'optimisation classiques, les algorithmes génétiques n'utilisent pas les variables mais leur associe un codage particulier, les variables de décision sont prises en compte à l'aide de caractères représentant une séquence de codes. Chaque variable de décisions est traduite sous forme d'un gène qui peut contenir un ou plusieurs codes, pouvant exprimer des caractères différents. La séquence de code représente un individu. En d'autre terme une solution potentielle connue sous le nom de chromosome. L'objectif du problème traité est exprimé à l'aide d'une fonction qui permet d'évaluer les chances qu'un individu soit sélectionné ou non pas, afin de reproduire de nouvelles solution. Cette fonction est la fonction d'adaptation (fitness). Les contraintes sont prises en compte dans la fonction d'adaptation en pénalisant les individus qui violent les contraintes du problème. L'exploration de l'espace des solutions possibles s'articule sur deux mécanismes qui cherchent à générer de manière aléatoire de nouvelles solutions à partir de la population de départ. Ces mécanismes sont les opérateurs génétiques : le croisement et la mutation. Le mécanisme de sélection cherche à diriger l'exploration en déterminant les individus ayant la plus grande probabilité d'être choisis. Afin d'expliciter le fonctionnement des algorithmes génétiques, nous présentons dans la suite de cette section les différentes étapes d'un algorithme génétique simple illustrées sur la Figure 2.1.

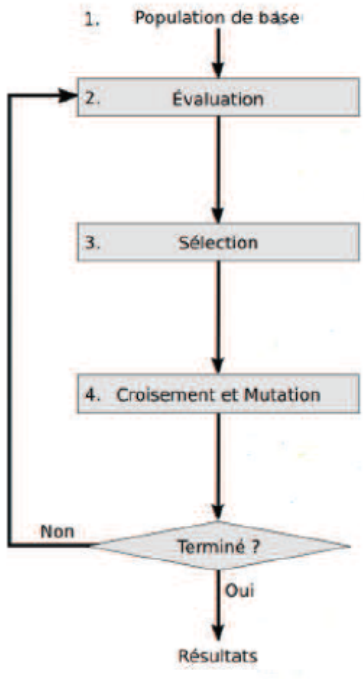

FIGURE 2.1 – Les principales étapes d'un algorithme génétique

2.1.2 Codage et population initiale

Un aspect important des algorithmes génétiques est la façon dont sont codées toutes les solutions. Les algorithmes génétiques établissent une analogie entre l'ensemble de solutions d'un problème et l'ensemble d'individus d'une population naturelle, en codant l'information sur chaque solution. En fonction du problème étudié, les codes utilisés peuvent être binaires ou réels. L'AG démarre avec une population initiale d'individus dans le codage retenu. Le choix des individus conditionne fortement la rapidité de l'algorithme. Si la position de l'optimum dans l'espace de recherche est totalement inconnue, il est intéressant que la population soit répartie sur tout l'espace de recherche. Si par contre des informations à priori sur le problème sont disponibles, il paraît évident de générer les individus dans un espace particulier afin d'accélérer la convergence. Disposant d'une population initiale souvent non homogène, la diversité de la population doit être entretenue aux cours des générations afin d'explorer le plus largement possible l'espace de recherche.

2.1.3 Évaluation de la population

Cette étape consiste à évaluer chaque solution contenue dans la population, la mesure de performance des solutions s'appuie sur la valeur de la fonction objectif. Cette étape permet de classer les solutions, afin de déterminer les solutions qui seront sélectionnées pour construire une nouvelle population de solutions.

2.1.4 Opérateur de sélection

La sélection a pour objectif d'identifier les individus qui doivent se reproduire. Cet opérateur ne crée pas de nouveaux individus mais identifie les individus sur la base de leur fonction d'adaptation, les individus les mieux adaptés sont sélectionnés alors que les moins bien adaptés sont écartés. La probabilité de sélectionner un individu donné est souvent traduite par le rapport entre la valeur de sa fonction d'adaptation et la somme de toutes les fonctions d'adaptation de la population. Il existe plusieurs techniques de sélection, nous en développons trois : la sélection par roulette biaisée (roulette wheel selection), la sélection par tournoi (tournement selection) et la sélection par rang (ranking selection).

Sélection par roulette

Selon cette méthode, chaque chromosome est copié dans la nouvelle population proportionnellement à sa fitness. On effectue en quelque sorte, autant de tirages avec remise que d'éléments existant dans la population. Ainsi pour un chromosome particulier ch_i de fitness $f(ch_i)$ la probabilité de sa sélection dans la nouvelle population de taille N est :

$$P\left(ch_i\right) = \frac{f\left(ch_i\right)}{\sum_{j=1}^{N} f\left(ch_j\right)} \tag{2.1}$$

FIGURE 2.2 – Sélection par roulette

Sélection par tournoi

La sélection consiste à tirer deux individus aléatoirement dans la population et on reproduit le meilleur des deux dans la nouvelle population. On répète la procédure jusqu'à'à ce que la nouvelle population soit complète.

Sélection par rang

Il s'agit de classer la population suivant la fonction d'adaptation, chaque individu de la population se voit accorder un rang. Plus l'individu est bon, plus son rang est élevé. Le principe de la sélection par rang est similaire à la sélection par roulette, la différence est que la proportion est calculée sur les rangs et non sur la valeur de la fonction d'adaptation. L'ensemble des individus est représenté sur un segment de droite dont les valeurs sont comprises entre 0 et 1.

2.1.5 Opérateur de croisement

Le croisement est le principal opérateur agissant sur la population des parents. Il permet de créer de nouveaux individus par l'échange d'information entre les chromosomes par leur biais de leur combinaison. La population courante est divisée en deux sous populations de même taille et chaque couple formé par un membre provenant de chaque sous population participe à un croisement avec une probabilité souvent supérieure à 0.5. Si le croisement a eu lieu entre deux chromosomes parents (ch1 et ch2), on tire aléatoirement une position de chacun des parents. On échange ensuite les deux sous chaînes terminales de chacun des chromosomes, ce qui produit deux enfants $ch_1^{'}$ et $ch_2^{'}$ comme indiqué sur la Figure 2.3.

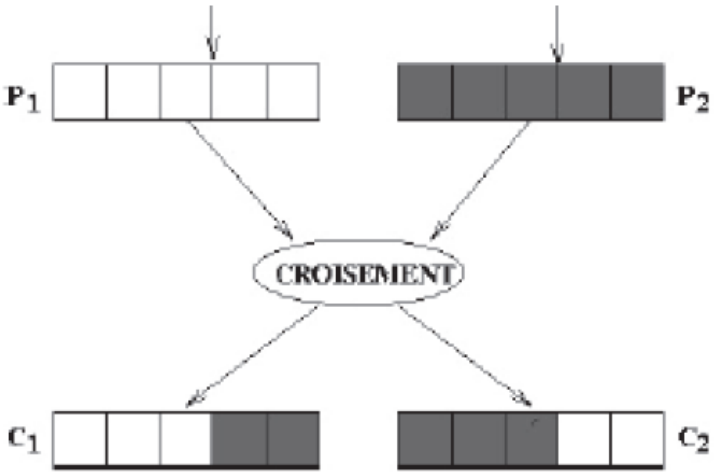

FIGURE 2.3 – Croisement standard en un seul point

où α est un paramètre de pondération aléatoire qui prend généralement ses valeurs dans l'intervalle [-0.5, 1.5] [4]

2.1.6 Opérateur de mutation

Le rôle de cet opérateur est de modifier aléatoirement la valeur d'un gène dans un chromosome. Dans le cas du codage binaire, chaque bit $a_i \in \{0, 1\}$ est remplacé par son complémentaire $a_i = 1 - a_i$. Dans l'exemple de la figure 2.6, une mutation a eu lieu sur le cinquième gène du chromosome ch et elle a transformé ce gène de 1 en 0.

Dans le cas d'un codage réel, on utilise principalement deux opérateurs de mutation : la mutation uniforme et la mutation non uniforme (Michalewicz,1992). En supposant fixée la probabilité de mutation p_m, un tirage au sort pour chaque gène x_k d'un chromosome ch permet de décider si ce gène doit être modifié ou non. Nous supposons que le gène prend ses valeurs dans un intervalle $[x_k^{min}, x_k^{max}]$.
Pour la mutation uniforme, qui est une simple extension de la mutation binaire, on remplace le gène x_k sélectionné par une valeur quelconque $x_k^{'}$ tirée aléatoirement dans l'intervalle $[x_k^{min}, x_k^{max}]$.
Pour la mutation non uniforme, le calcul de la nouvelle valeur d'un gène est un peu plus complexe. Le gène x_k subit des modifications importantes durant les premières générations, puis graduellement décroissantes au fur et à mesure que l'on progresse dans le processus d'optimisation. Pour une génération t, on tire au sort une valeur binaire qui décidera si le changement doit être positif ou négatif. La nouvelle valeur $x_k^{'}$ du gène x_k est donnée par :

$$x'_k \begin{cases} x_k + \Delta\left(t, x_k^{\max} - x_k\right) & \text{si rand } = 0 \\ x_k - \Delta\left(t, x_k - x_k^{\min}\right) & \text{si rand } = 1 \end{cases} \tag{2.2}$$

où $\Delta(t, y)$ est une fonction qui définit l'écart entre la nouvelle valeur et la valeur initiale à la génération t et rand est un nombre aléatoire qui prend les valeurs 0 ou 1.

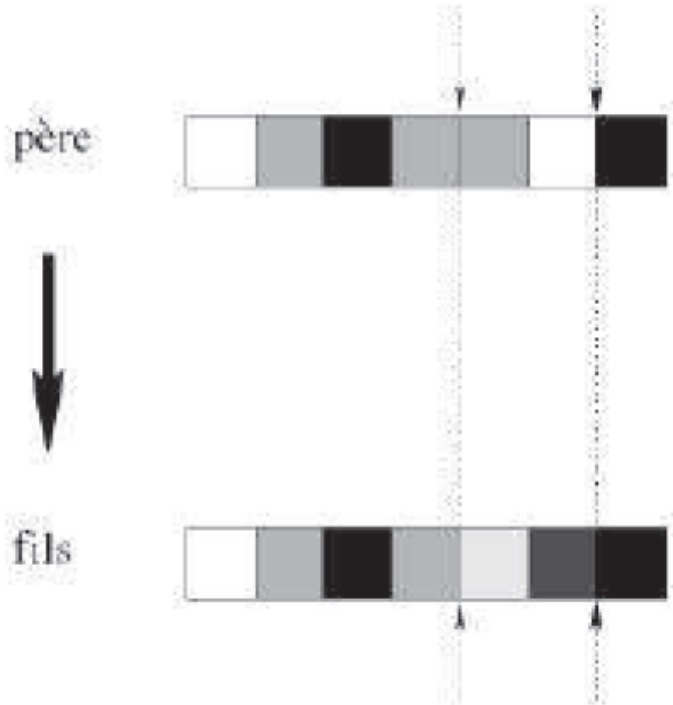

FIGURE 2.4 – Shéma de mutation

2.1.7 Cas pratique

Hello World !

Pour ce cas pratique nous allons utiliser un algorithme génétique afin de reproduire le texte "Hello World !". Bien que ce problème soit trivial, le but de ce cas pratique est de comprendre pas à pas comment utiliser et implémenter les algorithmes génétiques.

L'individu

Commençons par définir l'individu : dans notre cas celui-ci est composé que d'un seul chromosome. Un chromosome sera composé de plusieurs gènes représentant les lettres du texte, ainsi que de son coût :
Pour définir son coût, on définit une fonction effectuant le delta entre le but à atteindre et l'état actuel de l'individu, ou chromosome dans notre cas. Cette fonction sera utilisé par la fonction d'évaluation et de sélection.
Ensuite, on définit la fonction de croisement entre deux chromosome en effectuant un simple enjambement. Celle-ci retournera deux enfants, mais il est tout à fait possible de retourner un ou plusieurs enfants.
Enfin, on définit la fonction de mutation avec sa probabilité. Si la mutation survient, on choisit aléatoirement un gène/caractère. Sa mutation sera le replacement de la lettre sélectionnée par la suivante ou la précente dans l'alphabet ASCII.

La population

La population dans ce cas pratique est une collection de texte, avec un but à atteindre ("Hello World !") et une taille limite. Son initialisation est la Genèse.
Ensuite, on définit la fonction de sélection. Celle-ci utilisera une sélection par rang en ne sélectionnant que les deux tiers des individus les mieux adaptés. Pour ceci, nous avons besoin d'une fonction de tri qui effectuera une comparaison sur le score d'adaptation, ou le coût.
On souhaite maintenant pouvoir effectuer les opérations de croisement et de mutation sur l'ensemble de la population. - Pour le croisement, uniquement la moitié des individus se reproduiront avec un/une partenaire au hasard. - Pour la mutation, chaque individu aurra une probabilité de 70% d'être affecter.
Enfin, on définit la fonction de génération qui implémentera la boucle d'évolution de notre algorithme. Le programme dispose des composants nécessaires à la résolution du problème "Hello World !".
Voici un exemple de fonctionnement avec un temps d'attente de 200ms pour visualiser chaque étape de génération. Cliquez sur "Result" pour voir le résultat, la colonne de gauche est la représentation de l'individu et la colonne de droite son score.

2.1.8 Liste d'exemples

Voici une liste d'exemples utilisant les algorithmes génétiques :

- Cryptanalyse : Bruteforce pour obtenir la clef privée sur des clés asymétriques.

- Finance : Prédiction de l'évolution d'une action.

- Résolution des Sudokus : Obtenir la solution à un sudoku.

- Planning : Obtenir le meilleur planning en fonction de dispositions.

- Plan de table : Effectuer la meilleure disposition de table en fonction des affinités de chacun.

- Robotique : Comportement intuitif et apprentissage.

- Data Mining : Création et utilisation de règles pour obtenir de nouvelles informations.

2.2 Recuit Simulé

Le recuit simulé est une méthode empirique inspirée d'un processus utilisé en métallurgie, on présente dans cette partie la méthode d'optimisation du recuit simulé sur un espace fini. On étudiera qu'à un problème d'optimisation peut être associé avec une chaine de Markov qui se concentre sur l'ensemble des minima.

2.2.1 Définition

(Chaînes de Markov Homogènes).Soit $(U_n)_{n>0}$ une suite de variables aléatoires à valeurs dans un espace dénombrable E. Si pour tout entier $n \geq 0$ et tous états $i_0, i_1, ...i_{n-1}, i, j$

$$P(U_{n+1} = j \mid U_n = i, U_{n-1} = i_{n-1}, ..., U_0 = i_0) = P(u_{n+1} = j \mid U_n = i). \tag{2.3}$$

2.2.2 Principes de la méthode

Pour fabriquer un acier de bonne qualité on le recuit plusieurs fois en effectuant des fusions à des températures lentement décroissantes, cette procédure physique est reprise dans la méthode mathématique du recuit simulé. Le recuit simulé est une technique d'optimisation de type Monte-Carlo, c'est à dire une méthode visant à calculer une valeur numérique en utilisant des procédés aléatoires, cette méthode est le résultat d'expériences réalisées par Metropolis et collaborateurs dans les années 50 pour simuler l'évolution de ce processus de recuit physique.
L'idée heuristique du recuit simulé s'inspire de la technique expérimentale du recuit utilisée en métallurgie. Celle-ci permet d'obtenir un état stable, ou d'énergie minimale du métal. Cet état est obtenu quand le matériau a trouvé une structure cristalline ([11], [12], [13]). Par analogie avec le processus physique, on peut avoir le tableau suivant :

terme physique	terme mathématique
- état du système	- solution
- position moléculaire	- variable
- énergie	- fonction objectif
- état stable	- solution optimale globale
- état métastable	- solution optimale locale
- température	- paramètre de contrôle

2.2.3 L'algorithme du recuit simulé :

La valeur minimale globale d'une fonction $f : \mathbb{R}^n \to \mathbb{R}$ est atteinte quand le système est proche de son équilibre thermodynamique.
Comme l'algorithme de recuit simulé est un algorithme itératif qui construit la solution au fur et à mesure. Le déroulement de l'algorithme est le suivant (en notant k le numéro de l'itération) :

i. Tirer suivant la loi uniforme un pointx_1

ii. Etant donné x_k ; trouver un voisin y_k de x_k (choisir une loi de voisinage). Evaluer $f(y_k)$

iii. Si $f(y_k) \leqslant f(x_k)$; conserver y_k et prendre $y_k = x_{k+1}$:(c'est-à-dire transition accepté).
Sinon, conserver y_k avec probabilité $\exp(-\beta_k(f(y_k) - f(x_k))$avec β_k un paramètre positif (c'est-à-dire une transition selon une probabilité p_k).

iv. Vérifier ?er certaine critère d'arrêt (le taux de changement est faible) et retourner à l'étape (ii) L'algorithme du recuit simulé fonctionne comme suit :
Dans l'étape k, on a deux possibilités, si $f(y_k) < f(x_k)$; on sait que x_k n'est pas bon, on prend $y_k = x_{k+1}$: Pour ne pas être bloqué en un minimum local qui peut être x_k. On prend $y_k = x_{k+1}$ si $f(y_k) - f(x_k)$ est inférieur à une certaine variable aléatoire positive. Deuxième possibilité on prend $x_k = x_{k+1}$.
   — On tire au hasard un point y_k à partir de x_k par une relation de voisinage (une loi engendrée par une matrice de transition markovienne) :
   — Les itérations successives de voisinage (xk) construisent une chaine de Markov homogène (le mécanisme de transition ne change pas au cours du temps) convergeant vers la loi stationnaire de Gibbs $G_{f,T}$ qui dépend de $f(x)$ et T : Tant que la loi de Gibbs est associée à la fonction objectif et la température et sous certaine condition, la loi $G_{f,T}$ se concentre sur l'ensemble des minimiseurs globaux de $f(x)$ lorsque T tend vers

2.2.4 Convergence de l'algorithme

Sous certaines conditions, le recuit simulé converge sans doute vers un optimum global, dans le sens qu'on peut obtenir une solution proche de l'optimum. Pour plus de détails, voir [33].

2.2.5 Avantages et inconvénients

Le recuit simulé permet la recherche d'un minimum global. C'est son principal avan-tage, cette méthode a l'avantage d'être souple lorsqu'on veut résoudre des problèmes d'optimisation combinatoire, le plus souvent de grande taille.

Parmi les inconvénients du recuit simulé, des difficultés 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 para- mètres sont souvent choisis de manière empirique. D'autre part, la méthode est lente. C'est son principal inconvénient. Cet inconvénient est du reste commun à tous les algorithmes probabilistes de recherche de minimum global (recuit simulé, algorithmes génétiques,...)

2.2.6 Conclusion

Le recuit simulé est une méthode d'optimisation importante historiquement. Grâce à son implémentation simple et ses propriétés de convergence intéressantes, elle trouve son application dans de nombreux domaines dans lesquels on a à résoudre des problèmes d'optimisation difficiles, les chercheurs l'ont utilisée essentiellement dans :

— Le problème du voyageur de commerce ;
— La segmentation d'images ;
— La conception des circuits intégrés ;
— Le problème du sac à dos.

CONCLUSION GÉNÉRALE

Les problèmes d'optimisation globale sont extrêmement difficiles à résoudre, sont de type NP-difficile. Ils sont même intraitables sans un minimum d'hypothèses qui doivent être exigées sur la fonction objectif (différentiable, Lipschitz,...).

Dans le cas d'une fonction à une seule variable ces problèmes sont faciles à résoudre. Les difficultés deviennent beaucoup plus considérables lorsqu'on a besoin du minimum global d'une fonction à plusieurs variables, non convexe ou non linéaire.

Le premier chapitre est consacré à la mise au point d'une technique de généralisation directe des méthodes de Piyavskii et d' Evtushenko pour résoudre les problèmes d'optimisation globale des fonctions lipschitziennes dans le cas unidimensionnelles.Les deux facteurs qui agissent sur la convergence de ces méthodes sont le choix de la constante de Lipschitz et les calculs auxiliaires.

Enfin, le besoin énorme pratique exigeant la résolution des problèmes d'optimisation globale, ce qui nous incitent à chercher de nouvelles techniques de restitutions

Une autre voie consiste à utiliser ces techniques dans le cas unidimensionnelle des fonctions de Hölder.

Le second chapitre, est pratiquement consacrer aux méthodes stochastiques, ou bien aux méthodes appelées aussi évolutionnaires

BIBLIOGRAPHIE

[1] Contribution à l'optimisation globale : approche déterministe et stochastique et application Mohamed Zeriab Es-Sadek

[2] J. Holland. Adptation in natural and aritificial systems. University of Michigan Press, 1975.

[3] J. D. Knowles, D.W. Corne. Approximating the nondominated front using the pareto archived evolutionary strategy. Evolutionary Computation 8 :149–172, 2000.

[4] D.E. Goldberg. Genetic algorithms for search, optimization, and machine learning. In : Addison-Wesley, MA : (ed), Reading, 1989.

[5] C.A.C. Coello, E.M. Montes. Constraint-handling in genetic algorithms through the use of dominance-based tournament selection. Advanced Engineering Informatics, 16 :193–203, 2002.

[6] M. Rahal. Extension de certaines méthodes de recouvrement en optimisation globale, Thèse de doctorat de l'Université Ferhat Abbas Setif (2009).

[7] Gourdin E.,Jaumard B.,and Ellaia R.,Global optimization of Holder funtions,Journal of Global Optimization 8,323-348(1996).

[8] Yu. G.Evtushenko. Numerical methods for finding global extreme (case of a non-uniform mesh). U.S.S.R. Comput. Maths. Math. Phys., Vol. 11, N 6, pp. 38-54, 1971.

[9] Yu. G.Evtushenko. Numerical Optimization Techniques, Optimization Software. New York : Inc. Publications Division, p. 567, 1985.

[10] Yu. G. Evtushenko, V. U. Malkova, and A. A. Stanevichyus. Parallelization of the Global Extremum Searching Process. Automation and Remote Control. Vol. 68, No 5, Pp. 787-798, 2007.

[11] P. Siarry et Rahoual. Méthodes et pratiques de l'ingénieur. In Réseaux informa- tiques : Conception et Optimisation.

[12] Berthiau Gérard. La méthode du recuit simulé pour la conception des circuits électroniques : adaptation et comparaison avec d'autres méthodes d'optimisa- tion . PhD thesis, Ecole centrale des arts et manufactures, Chatenay-Malabry, FRANCE, 1994.

[13] Fieguth.P et Lee.L.J Jamieson.M. Parametric contour estimation by simulated annealing. ICIP03, 3 :449 452, 2003.

[14] Jaumard B. and Ellaia R. Gourdin, E. ”global optimization of holder function”. Journal of Global Optimization, 8(4) :323–348, (1996).

[15] D.Lera and Ya.D.Sergeyev. ”lipschitz and holder global optimization, usingspace ?lling curves”. Bit Issn 0006-3835 Coden Nbitab, 42(1) :119–133, (2002).