Contribution à l'optimisation globale : approche déterministe et stochastique et application

Mohamed Zeriab Es-Sadek

Université Mohammed V - Agdal
École Mohammadia d'Ingénieurs
Laboratoire d'Étude et Recherche en Mathématiques Appliquées
Rabat - Maroc

Institut National des Sciences Appliquées - Rouen
Laboratoire de Mécanique de Rouen
Rouen - France

THÈSE EN COTUTELLE

présentée à

L'ÉCOLE MOHAMMADIA D'INGÉNIEURS

pour obtenir le grade de

DOCTEUR EN SCIENCES APPLIQUÉES

et

L'INSA de Rouen

pour obtenir le grade de

DOCTEUR DE L'INSA DE Rouen

Mention : MATHÉMATIQUES | Équipe d'acceuil : LMR

par

Mohamed Zeriab ES-SADEK

CONTRIBUTION A L'OPTIMISATION GLOBALE. APPROCHE DÉTERMINISTE ET STOCHASTIQUE ET APPLICATION

Soutenue le **21 Novembre 2009** devant la commission d'examen

M. :	Abdelkhalek EL HAMI	INSA, Rouen	Président
M. :	Youssef BENADADA	ENSIAS, Rabat	Rapporteur
M. :	Piotr BREITKOPF	U. T, Compiègne	Rapporteur
M. :	Mohamed TKIOUAT	EMI, Rabat	Rapporteur
M. :	Omar KAITOUNI IDRISSI	EMI, Rabat	Éxaminateur
M. :	Rachid ELLAIA	EMI, Rabat	Directeur de thèse
M. :	José Eduardo SOUZA DE CURSI	INSA, Rouen	Directeur de thèse

REMERCIEMENTS

Permettez moi de citer nominativement, ces personnes qui à mes yeux sont les substrats de mon essence scientifique et morale, afin de les remercier en ces quelques lignes :

A tous les membres de ma famille et plus spécialement mon Père, ma Mère et ma Soeur,

Monsieur le Président du jury :
Professeur El Hami Abdelkhalek, d'un profond respect de soi et des autres et d'une humilité à l'image de sa prestance,

Messieurs les encadrants :
Professeur Ellaia Rachid , encadrant d'une grande disponibilité et d'une valeur encyclopédique incommensurable,
Professeur Souza De Cursi José Eduardo, encadrant d'une clairvoyance et d'une pertinence rare,

Messieurs les Rapporteurs :
Professeur Tkiouat Mohamed,
Professeur Benadada Youssef,
Professeur Breitkopf Piotr,

Monsieur l'examinateur :
Professeur Idrissi Kaitouni Omar,

Tous les membres du laboratoire d'étude et de recherche en mathématiques appliquées à l' E.M.I. et tous les membres du laboratoire de mécanique de l'I.N.S.A. de Rouen ;

Suite à l'achèvement de notre thèse intitulée : CONTRIBUTION A L'OPTIMISATION GLOBALE, APPROCHE DETERMINISTE ET STOCHASTIQUE ET APPLICATION, nous tenons à mettre l'accent que sous votre supervision et votre collaboration, nous avons prôné plusieurs idées motrices : principe d'une meilleure valeur ajoutée.

Votre mobilisation, vos directives et suggestions ont été garantes de l'atteinte de l'objectif escompté.

Et ce en plus, nous tenons à souligner, par la même occasion que cet ensemble synergétique, a enclenché en nous une nouvelle dynamique, que nous tenons à faire consolider en vous proposant ce travail contributif.

Nous saisissons cette opportunité, pour nous rappeler à jamais que l'ensemble de vos interventions pertinentes, ont été pour ma personne un mémorandum d'une coopération fructueuse et prometteuse.

Le volume, d'échange et d'idées, bilatéral, qui tranche, par son objectivité, nous a amené à porter plus nos critères, que partage nos aspirations dans un éternel respect mutuel .

Je vous réitère, encore une fois, mes sincères remerciements et vous souligne notre éternel attachement aux méthodes, processus et déontologie qui, force d'acharnement de votre part, vous avez tenu à m'inculquer profondément.

Je remercie l'institution CMIFM (Egide, PAI Volubilis, MA/07/173) pour son financement de mes stages au sein de l'I.N.S.A. de Rouen.

Abstract

This thesis concerns the global optimization of a non convex function under non linear restrictions, this problem cannot be solved using the classic deterministic methods like the projected gradient algorithm and the sqp method because they can solve only the convex problems. The stochastic algorithms like the genetic algorithm and the simulated annealing algorithm are also inefficients for solving this type of problems. For solving this kind of problems, we try to perturb stocasicly the deterministic classic method and to combine this perturbation with genetic algorithm and the simulated annealing.

So we do the combination between the perturbed projected gradient and the genetic algorithm, the perturbed sqp method and the genetic algorithm, the perturbed projected gradient and the simulated annealing, the Piyavskii algorithm and the genetic algorithm. We applicate the coupled algorithms to different classic examples for concretited the thesis. For illustration in the real life, we applicate the coupled perturbed projected gradient end the genetic algorithm to logistic problem eventuelly transport. In this view, we sold the efficient practices.

Keywords

Global optimization, stochastic perturbation, projected gradient, sqp, genetic algorithm, simulated annealing, hybridation, logistic problem.

RÉSUMÉ

Dans les situations convexes, le problème d'optimisation globale peut être abordé par un ensemble de méthodes classiques, telles, par exemple, celles basées sur le gradient, qui ont montré leur efficacité en ce domaine. Lorsque la situation n'est pas convexe, ces méthodes peuvent être mises en défaut et ne pas trouver un optimum global.

La contribution de cette thèse est une méthodologie pour la détermination de l'optimum global d'une fonction non convexe, en utilisant des algorithmes hybrides basés sur un couplage entre des algorithmes stochastiques issus de familles connues, telles, par exemple, celle des algorithmes génétiques ou celle du recuit simulé et des algorithmes déterministes pertubés aléatoirement de façon convenable. D'une part, les familles d'algorithmes stochastiques considérées ont fait preuve d'efficacité pour certaines classes de problèmes et, d'autre part, l'adjonction de perturbations aléatoires permet de construire des méthodes qui sont – en théorie – convergents vers un optimum global. En pratique, chacune de ces approches a ses limitations et insuffisantes, de manière que le couplage envisagé dans cette thèse est une alternative susceptible d'augmenter l'efficacité numérique. Nous examinons dans cette thèse quelques unes de ces possibilités de couplage. Pour établir leur efficacité, nous les appliquons à des situations-test classiques et à un problème de nature stochastique du domaine des transports.

Mots clés

Optimisation globale, perturbation aléatoire, gradient projeté, sqp, algorithme génétique, recuit simulé, hybridation, problème logistique.

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction générale

1.1 Optimisation d'un dispositif

Lorsqu'on souhaite construire un dispositif quelconque, un des premiers problèmes à résoudre est celui de l'organisation générale et du dimensionnement de ses composants. On aboutit ainsi à la nécessité de définir des procedures cohérentes et reproductibles pour la détermination de valeurs, position, côtes, matériaux ou architectures.

Dans une première approche, on peut se baser sur une méthode d'essai et d'erreurs, où la solution passe par diverses étapes d'expérimentation potentielles jusqu'à obtention de résultats adéquats. Des exemples simples de cette méthode sont, par exemple

- la variation à la main d'un paramètre par exemple.
- Le traçage de la courbe (cf. figure 1.1), est l'expression schématique de la variation quasi-continue de ce paramètre.

La méthode d'essai et d'erreur s'emploie dans le contexte général où la solution est inconnu. On peut l'améliorer en éliminant des solutions peu fiables. Cette démarche fait intervenir des procedures d'optimisation et de résolution de problèmes.

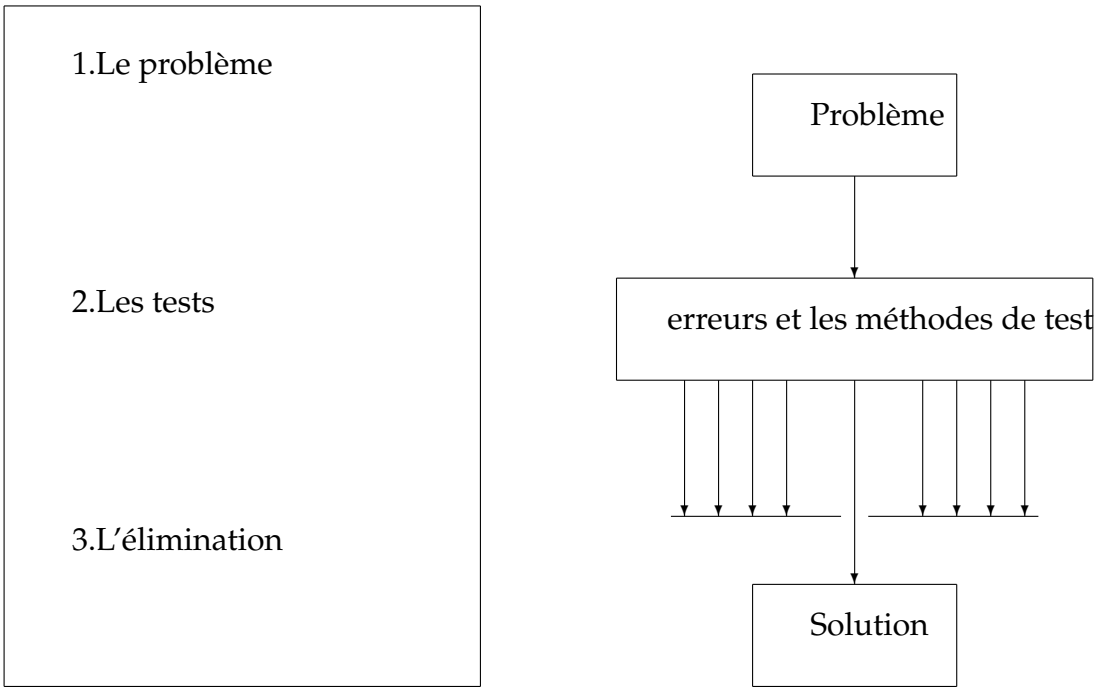

FIGURE 1.1 – Apprentissage par essai et erreur par un schéma à trois niveaux.

Le raisonnement à Partir de Cas (RàPC) fait partie des outils d'intelligence artificielle couramment utilisés pour la réalisation de systèmes à base de connaissances pour aider les concepteurs.
De nombreuses contributions aux systèmes de RàPC ont permis une comparaison avec d'autres systèmes tels que le raisonnement à partir de règles ou encore à partir de modèles. En effet, la capitalisation des connaissances sous forme de cas est plus immédiate que d'utiliser d'autres formes de modélisation telles que les règles ou les contraintes. Les systèmes à partir de cas sont plus performants autant en termes d'application que de précision, tout du moins pour les problématiques de planification, de diagnostique et de conception. Si l'on compare la méthode à partir de cas avec un système à base de règles ayant la même fonctionnalité, sa construction nécessite moins de personnes et de temps. La force de la première méthode par le fait que ce sont les cas qui sont capitalisés, est qu'elle privilégie le contenu de la connaissance ; toutefois, la grande difficulté est l'indexation des cas. Dans la deuxième, la connaissance provenant dexperts est encodée en règles.

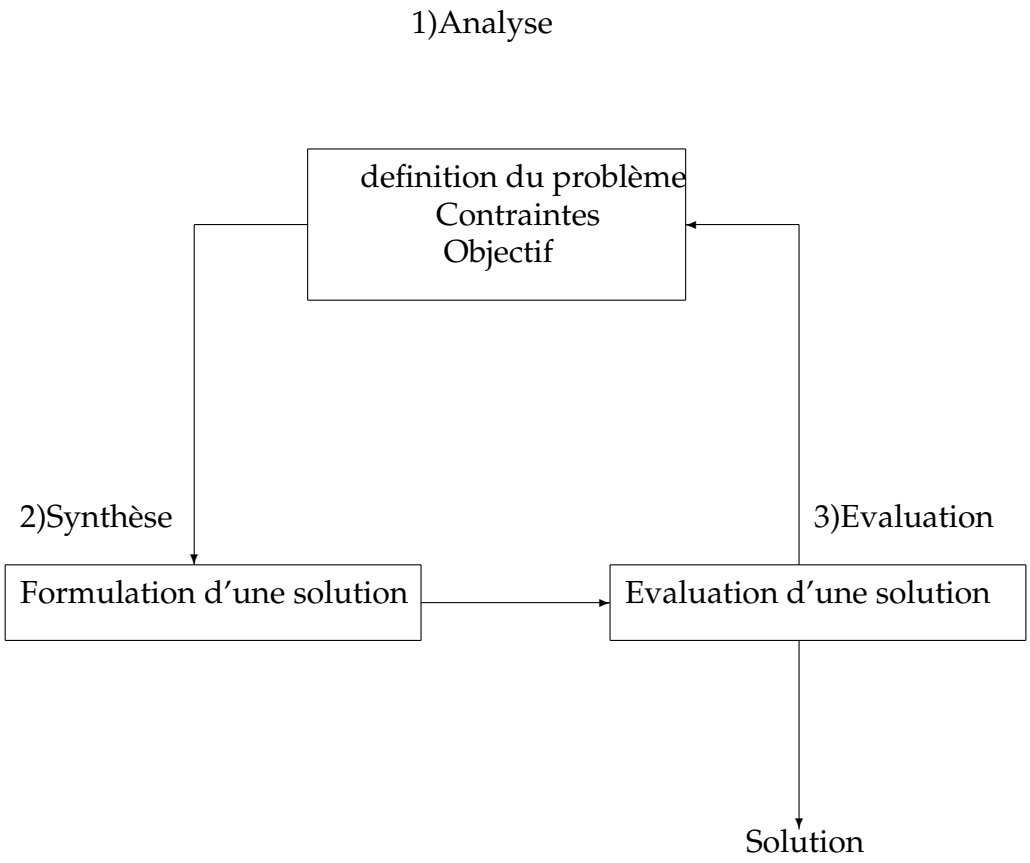

FIGURE 1.2 – Processus d'optimisation selon Asimow.

Une seconde approche de la même famille est celle de TRIZ, qui est l'acronyme russe pour "Téoria Rechénia Izobrétatelskikh Zadatch", traduit par "Théorie de la Résolution des Problèmes Inventifs". TRIZ a été créée puis développée par Genrich Altshuller en Ex URSS. TRIZ peut se définir comme un outil de génération d'idées dans le cadre de projets innovants.
TRIZ se différencie des méthodes de groupe par son côté systématique : des concepts de solution sont puisés dans des bases de données pluridisciplinaires élaborées à partir de savoirs industriels capitalisés et structurés. Pour naviguer dans ces bases de données, les problèmes rencontrés sont modélisés par des problèmes génériques pour lesquels des solutions génériques existent.

D'un point de vue pratique, le processus d'optimisation se décompose en 3 étapes :

L'analyse, la synthèse et l'évaluation sont les trois étapes du processus
d' évaluation comme il est shématisé dans la figure 1.2. L'analyse du problème est en substance la réponse à un certain nombre de choix préalables :

- Variables du problème. Quels sont les paramètres intéressants à faire varier ?
- Espace de recherche. Dans quelles limites faire varier ces paramètres ?
- Fonctions objectif. Quels sont les objectifs à atteindre ?
- Méthode d'optimisation. Quelle méthode choisir ?

La synthétisation de la méthode choisie, permet l'évolution de la solution trouvée voire son élimination jusqu' à obtention d'une résultante acceptable.
Si nécessaire , le problème peut être redéfini à partir des solutions déjà obtenues.

Variables du problème

Les variables peuvent être de natures diverses.
Par exemple, pour un composant électronique il peut s'agir de sa forme et ses dimensions géométriques, des matériaux utilisés, des conditions de polarisation, etc.
Leurs définition passe par un choix empirique. c'est à l'utilisateur de le définir. Il peut avoir à faire varier un grand nombre de paramètres afin d'augmenter les degrés de liberté de l'algorithme.
Par la suite nous désignerons par $x_1, \ldots, x_n$ les n variables du problème. Celles-ci peuvent être réelles, complexes, entières, booléennes, etc. Ici nous les supposerons réelles.

Espace de recherche

Dans certains algorithmes d'optimisation, tels que les Stratégies d'Evolution, la norme de l'espace de recherche est infini, en effet seule la population initiale est confinée dans un espace fini [5].
Mais dans le cas des algorithmes de type Monte Carlo et génétique, il est généralement nécessaire de définir un espace de recherche fini. Cette limitation de l'espace de recherche n'est généralement pas problématique. De ce fait, pour des raisons technologiques ou informatiques (taille de la fenêtre de modélisa-

tion), les intervalles de définition des variables sont en général limités.
De plus, une idée des ordres de grandeur de variables du problème est généralement fournie.

1.2 Fonctions objectif et fonction d'adaptation

Les grandeurs à optimiser peuvent être par exemple une consommation, un rendement, un facteur de transmission, un profit, la faisabilité technologique, un coût, une durée de développement, etc.
Un algorithme d'optimisation nécessite généralement la définition d'une fonction rendant compte de la pertinence des solutions potentielles, à partir des grandeurs à optimiser.
C'est la fonction d'adaptation f ou fitness function en terminologie anglo-saxonne. La convergence de l'agorithme vers l'optimum de cette fonction f, quelle que soit sa définition en est la principale conséquence. Ainsi, la pertinence de la solution est un enjeu de la question posée à l'ordinateur. La fonction f est la traduction en language mathématique du désir de l'utilisateur.

Objectif unique

Dans le cas d'un objectif unique, l'aisance de la définition de la fonction d'adaptation de la fonction f en est généralement la conséquence.
Par exemple, si l'on se fixe l'objectif de trouver un dispositif dont le rendement est maximum, cette fonction sera égale au rendement. Le calcul de la fonction d'adaptation se fait en deux étapes. On commence par évaluer les caractéristiques des solutions potentielles en utilisant le modèle. Puis on calcule la fonction d'adaptation à partir de ces caractéristiques.

Objectifs multiples

Les problèmes d'optimisation doivent souvent satisfaire des objectifs multiples, dont certains sont concurrents. Une méthode classique consiste à définir des fonctions objectif f_i , traduisant chaque objectif à atteindre, et de les combiner au sein de la fonction d'adaptation. On établit ainsi un compromis. Le plus

simple est de se restreindre à une somme pondérée des fonctions objectif :

$$f = \sum_i \alpha_i f_i$$

où les poids α_i doivent être tels que la fonction d'adaptation reste bornée dans l'intervalle $[0, 1]$. Remarquons que certains α_i peuvent être négatifs afin de tenir compte de certaines contraintes du problème. C'est à l'utilisateur de fixer convenablement les poids α_i. On peut souvent classer les objectifs par importance mais les poids seront généralement adaptés par tâtonnement, jusqu'à l'obtention d'une solution acceptable. Malgré l'automatisation du processus d'optimisation, l'utilisateur doit paramétrer à la main la définition de la fonction d'adaptation.
Or, si dans la littérature les algorithmes sont analysés en profondeur, ce problème délicat est souvent marginalisé, car l'on se situe au sein des recherches actuelles en optimisation.
A la place d'une somme, on peut également utiliser un produit de type :

$$f = \prod_i f_i^{\beta_i}$$

ou des expressions plus complexes.
Il faut néanmoins être conscient des effets d'une telle combinaison des objectifs. En effet, deux solutions potentielles dont les fonctions objectif n'ont pas la même valeur peuvent aboutir à une même valeur de la fonction d'adaptation. De plus, un algorithme utilisant une telle approche ne convergera que vers une seule solution alors qu'il existe peut-être toute une famille de solutions remplissant les objectifs fixés. L'optimisation à objectifs multiples est un domaine de recherche en dynamique actuellement, face aux enjeux économiques et industriels. Des concepts tels que les niches écologiques ou l'optimalité de Pareto semblent prometteurs pour la résolution de ce genre de problème.

1.3 Méthodes d'optimisation

Après définition de la fonction objectif, le choix d'une méthodologie adaptée s'impose pour la résolution du problème.
Les méthodes d'optimisation, répondent à deux classements distincts :

- méthodes déterministes.
- méthodes non déterministes.

Méthodes déterministes

l'efficacité des méthodes déterministes s'articule autour de la rapidité de l'évaluation de la fonction ou selon la connaissance à priori de ladite fonction. La recherche des extrema d'une fonction f revient à résoudre un système de n équations à n inconnues, linéaire ou non :

$$\frac{\partial f}{\partial x_i}(x_1, \ldots, x_n) = 0.$$

On peut donc utiliser des méthodes classiques telles que la méthode du gradient ou la méthode de Gauss-Seidel [12, 38]. En général, l'utilisation de ces méthodes nécessite comme étape préliminaire la localisation des extrema. Celle-ci peut être faite, par exemple, sur un graphique ou par une discrétisation fine de l'espace de recherche. La fonction à optimiser est évaluée en chacun des points de discrétisation. La valeur maximale est alors considérée comme une bonne approximation de l'optimum de la fonction. Cette méthode est brutale et le temps de calcul augmentera exponentiellement en fonction du nombre de variables. En effet, considérons une optimisation sur huit variables. Si on discrétise l'intervalle de définition de chaque variable en seulement 3 points, une exploration systématique nécessite $3^8 = 6561$ exécutions du modèle. Dans le cas d'un modèle physique nécessitant 10 secondes de calcul sur un PC standard, cela représente 18 heures de calcul, pour un résultat inutilisable !

Méthodes non déterministes

L'application des méthodes non déterministes pour des cas complexes en reflète le pragmatisme, tel que, par exemple :

- Temps de calcul important.
- Nombreux optimums locaux.
- Fonctions non dérivables.
- Fonctions fractales.
- Fonctions bruitées.

Ces méthodes font appel à des tirages de nombres aléatoires. Elles permettent d'explorer l'espace de recherche de manière plus exhaustive. Citons entre autres [6] :

- Les méthodes Monte Carlo : la fonction est évaluée en un grand nombre de points choisis aléatoirement.
- Les méthodes hybrides : on peut par exemple utiliser la méthode des gradients en partant d'un grand nombre de points choisis aléatoirement. On peut ainsi espérer déterminer au fur et à mesure tous les optima locaux de la fonction.
- Le recuit simulé : on effectue des déplacements aléatoires à partir d'un point initial. Si un déplacement mène à une valeur plus optimale de la fonction f, il est accepté. Sinon, la probabilité s'impose :

  $$p = e^{-\frac{|\Delta f|}{kT}}$$

  où "Δf" est la variation de la fonction, T est assimilé à une température qui décroît au cours du temps et k est une constante. Cette méthode est basée sur une analogie avec les processus de recuit utilisés en métallurgie et qui visent à atteindre une configuration d'énergie minimale ($-f$ est alors assimilée à une énergie).
- Les Algorithmes Evolutionnaires : le principe est de simuler l'évolution d'une population d'individus divers auxquels on applique différents opérateurs génétiques où l'on soumet à chaque génération à une sélection propre.
  Ces algorithmes sont de plus en plus utilisés dans l'industrie car ils sont particulièrement adaptés aux problèmes d'optimisation comportant de nombreux paramètres.

1.4 Compromis exploration et exploitation

Si nous opposons exploration et exploitation de l'espace de recherche, nous pouvons dire que les méthodes Monte Carlo amènent une bonne exploration. Elles permettent, de par leur nature, l'accés à tout point issu d'une probabi-

lité identique. Cependant, l'exploitation de résultats anterieurs est nulle et non-avenue.
Avec la méthode des gradients, l'exploration est moindre mais l'exploitation des données précédentes par leur intermédiaire permet une bonne recherche locale. Enfin, les Algorithmes Evolutionnaires offrent un bon compromis entre exploration et exploitation [7]. En ce sens, nous nous permettons d'aborder ce phénomène de manière plus détaillée.

1.5 Sujet de thèse

Suite à nos diverses notes explicatives, il nous est permis de constater que les problématiques réelles se voient modélisées à travers les méthodologies d'optimisation globale.
On peut citer, en guise d'exemple, les problèmes de :

- Calibration de modèles, souvent trouvées dans la logistique, l'économie ou l'identification, ...
- Design optimal des systèmes, souvent trouvé dans la mécanique.
- Analyse des problèmes d'agriculture.

A titre d'exemple, on considère dans cette thèse un problème logistique décrit dans le chapitre 8.

Dans toutes ces situations, on doit résoudre le problème modélisé suivant :

$$x^* = \operatorname{argmin} \{J(x) : x \in C\} \tag{1.1}$$

où $J : I\!R^n \longrightarrow I\!R$, $n \in I\!N$ est une fonction régulière et C est un ensemble non vide régulier, nommé l'ensemble des solutions réalisables. C est définie par des égalités et des inégalités :

$$C = \{x \in I\!R^n \,/\, h_i(x) \leq 0,\, i = 1, \ldots, p \quad \text{et} \quad h_i(x) = 0,\, i = p+1, \ldots, q\}. \tag{1.2}$$

Dans la suite, on suppose que les fonctions $h_i : I\!R^n \longrightarrow I\!R,\ i = 1,\ \ldots,\ q$ sont régulières.

Le problème 1.1 est appelé *un problème d'optimisation avec contraintes*.

L'optimisation avec contraintes pose des difficultées comparée aux problèmes sans contraintes. Pour résoudre cette difficulté, on peut trouver dans la litterature deux approches de base : la pénalité et la projection.

Dans l'approche basée sur la pénalité, on accepte des points non réalisables comme solutions approchées, souvent un paramètre de pénalité $\beta > 0$ est introduit pour générer une suite $\{X_\beta\}_{\beta>0} \subset I\!R^n$ de solutions approchées tel que $X_\beta \to X$ pour $\beta \to +\infty$ et $X \in C$, même si $X_\beta \notin C$ pour $\beta > 0$ [13].

L'avantage principal de l'approche de pénalité est sa simplicité, car elle trouve une approximation de l'optimum en utilisant des problèmes sans contraintes. Son inconvénient majeur est qu'elle peut générer des approximations non réalisables de la solution.

Les méthodes de projection tentent de générer une suite de points convergeant vers la solution : $\{X_i\}_{i\in I\!N} \subset C$ tel que $X_i \to x^*$. A chaque $i \in I\!N$, il est souvent nécessaire de considérer la projection de la direction sur un ensemble réalisable [50]. L'inconvénient de ce type de méthodes est qu'il requiert un temps de calcul très important.

Une autre difficulté se montre quand J ou C sont non convexes. Dans ce cas, la convergence globale n'est pas asurrée.

Pour résoudre ce type de problèmes, plusieurs approches ont été considérées dans la litterature : utilisation des conditions d'optimalité au lieu des méthodes de descente, méthode de relaxation pour approcher la solution en utilisant une suite de problèmes convexes ([33]), méthodes d'intervalles [24], approches heuristiques (recuit simulé [51], algorithme génétique [22], ...), méthodes déterministes [34], perturbation [58].

Dans ce travail, on considère une approche basée sur la perturbation stochastique de deux types de méthode : une méthode de pénalité qui est la méthode SQP, et une méthode de projection appelée le gradient projeté.
En fait, après avoir généré un point initial $X_0 \in C$, on applique soit la méthode SQP soit l'algorithme du gradient projeté puis on perturbe le point obtenu en utilisant un terme stochastique Q_k, et pour être sûre que le point perturbé est réalisable on le corrige vers la frontière de C en utilisant une méthode de correction définie dans le chapitre 3.

La perturbation Q_k est

$$Q_k = \xi_k Z_k,$$

où ξ_k est un paramètre positif et Z_k est une variable aléatoire.

Cette approche n'est pas toujours efficace pour trouver l'optimum global, nous amenant à utiliser une hybridation de la méthode SQP perturbée et de l'algorithme du gradient projeté perturbé, en couplant ces derniers avec des algorithmes stochastiques notamment le recuit simulé et l'algorithme génétique.

Ainsi, la première partie de cette thèse sera consacrée aux algorithmes détérministes et leurs perturbation :
Dans le chapitre 2 on rappelera un algorithme déterministe pour les fonctions Lipschitziennes qui est l'algorithme de Piyavskii.
Dans le chapitre 3 on parlera de l'algorithme du gradient projeté perturbé.
Dans le chapitre 4 on exposera la méthode SQP perturbée.
Dans la deuxième partie on définera des algorithmes évolutionnaires :
Dans le chapitre 5 on introduira l'algorithme génétique.
Dans le chapitre 6 on présentra le recuit simulé.
Le chapitre 7 sera celui de l'hybridation où on couplera l'algorithme de Piyavskii, l'algorithme du gradient projeté perturbé et la méthode SQP avec l'algorithme génétique, ainsi que l'algorithme du gradient projeté perturbé avec le recuit simulé.
Dans tous les chapitres des résultats numériques seront donnés en utilisant des fonctions test classiques.
Par suite, en guise d'application, un problème de transport sera définie et modélisé en un problème d'optimisation. L'hybridation sera appliquée à ce problème pour montrer son efficacité.

Première partie

Algorithmes déterministes et perturbation

Chapitre 2

Algorithme de Piyavskii et extension

2.1 Introduction

Dans ce chapitre, on considère le problème de maximiser une fonction f Lipschitzienne sur un compact X, c.à.d une fonction qui satisfait la condition de Lipshitz

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

où L est une constante appelée la constante de Lipshitz. 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.
Par exemple, le livre de Horst et Tuy [29] discute généralement les algorithmes déterministes. Piyavskii [43] propose une méthode déterministe séquentielle qui résoud ce problème en construisant itérativement une fonction borne supérieure F de f et en évaluant f en un point où F est maximale, Shubert [56], Basso [2, 3], Schoen [54], Shen et Zhu [55] et Horst et Tuy [30] proposent une autre formulation de l'algorithme de Piyavskii. Des extensions multidimensionelle ont été élaborées par Danilin et Piyavskii [14], Mayne et Polak [35], Mladineo [37] et Meewella et Mayne [36]. Evtushenko [17], Galperin [19] et Hansen, Jaumard et Lu [25, 26] proposent d'autres algorithmes pour ce type de problèmes où leurs extension au cas multidimensionel. Hansen et Jaumard [27] résument et discutent les algorithmes proposés et les présentent dans un language informatique.
Dans ce chapitre, on va présenter l'algorithme de Pijavskii qui est un algorithme

pour optimiser une fonction d'une seule variable sur un interval $[a, b]$, puis on va définir l'extension de cet algorithme au cas multidimensionnel, des résultats numériques sont aussi donnés pour montrer l'efficacité de cet algorithme.

2.2 Algorithme de Piyavskii

2.2.1 Introduction et description

L'algorithme de Piyavskii [43] maximise une fonction f Lipschitzienne définie sur un compact X, en construisant une fonction borne supérieure de f, $g : X \times X \to I\!R$ dont les propriétées sont les suivantes :

$$f(x) \leq g(x, y) \quad \forall x, y \in X \tag{2.1}$$

$$f(x) = g(x, x) \quad \forall x \in X \tag{2.2}$$

Si on a un ensemble de points $\{x_j, j = 0, \ldots, n\} \subseteq X$, la fonction borne supérieure est définie par :

$$G^n(x) = \min_{j=0,\ldots,n} g(x, x_j)$$

L'algorithme définie un suite $x_0, \ldots, x_n$ comme suit :
Soit x_0 un point dans X
A l'itération n on définie :

$$G^n(x) = \min_{j=0,\ldots,n} g(x, x_j)$$

$$M_n = \max_{x \in X} G^n(x)$$

x^{n+1} est un point qui satisfait $G^n(x_{n+1}) = M_n$
et on définie la suite $(f_n), n \geq 0$ par :

- $f_0 = f(x_0)$
- $f_{n+1} = \max\{f_n, f(x_{n+1})\}$

L'algorithme se termine quand

$$M_n - f_n < \epsilon$$

où ϵ est une tolérance choisie convenablement.

2.2.2 L'algorithme de Piyavskii-Shubert

Soit f une fonction qui satisfait la condition de Lipshitz sur un intervalle [a,b] :

$$f(x) - f(y) \leq L|x - y| \quad \forall x, y \in [a, b]$$

Soit $x_k \in [a, b]$, la fonction $g(x, x_k)$ est :

$$g(x, x_k) = f(x_k) + L|x - x_k| \quad \forall x \in [a, b]$$

g satisfait les conditions 2.1 et 2.2.

Algorithme

$k \leftarrow 1$;
$x^1 \leftarrow \dfrac{a+b}{2}$;
$x_{opt} \leftarrow x^1$;
$f_{opt} \leftarrow f(x_{opt})$;
$F_{opt} \leftarrow f_{opt} + \dfrac{L(b-a)}{2}$
$F_1(x) \leftarrow F(x) + L|x - x_1|$;
Tant que $F_{opt} - f_{opt} > \epsilon$ *faire*
  $x_{k+1} \leftarrow \max_{x\in[a,b]} F_k(x)$;
    Si $f_{opt} < f(x_{k+1})$;
      $f_{opt} \leftarrow f(x_{k+1})$;
      $x_{opt} \leftarrow x_{k+1}$;
    Fin Si ;
  $F_{k+1}(x) \leftarrow \min_{i=1,\ldots,k+1} f(x_i + L|x - x_i|)$
  $F_{opt} \leftarrow \max_{x\in[a,b]} F_{k+1}(x)$
  $k \leftarrow k + 1$
Fin Tant que

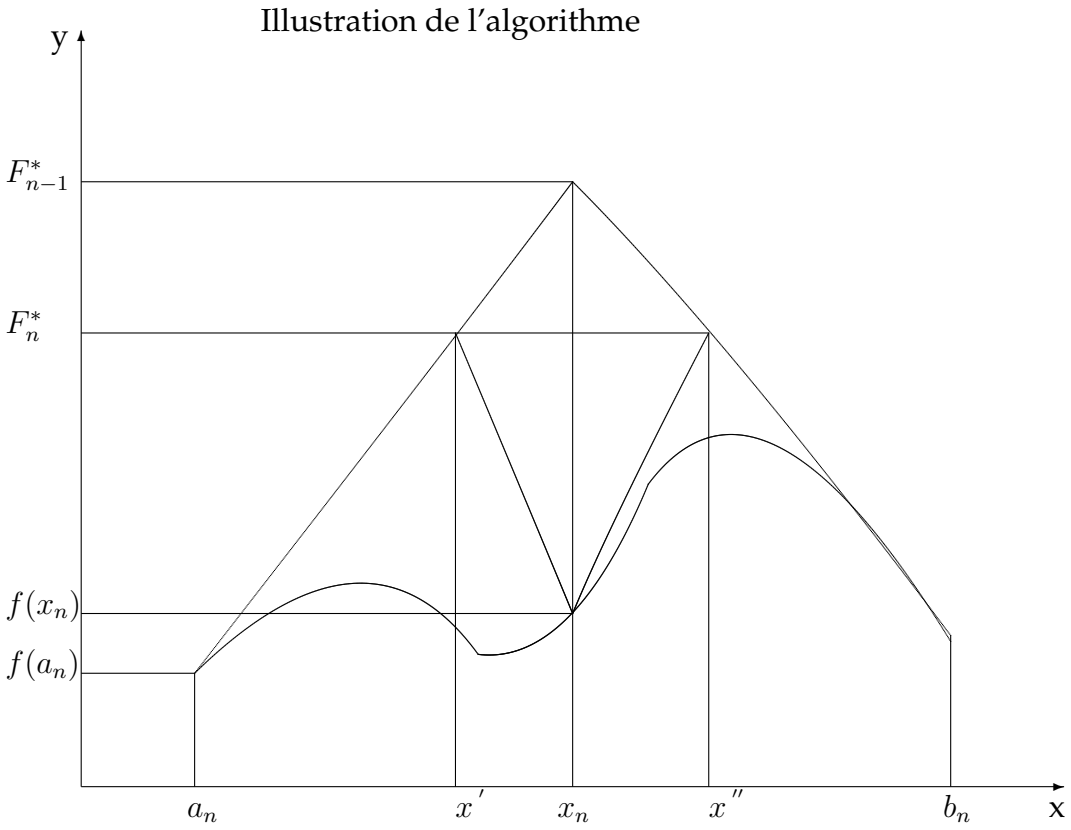

FIGURE 2.1 – Recouvrement de Piyavskii-Shubert

2.3 Optimisation multidimensionnelle

2.3.1 Optimisation sans contraintes

Considérons le problème :

$$\begin{cases} \text{Max } f(x) \quad x \in X \\ \text{avec } X = [a_1, b_1] \times \ldots \times [a_n, b_n] = [a, b] \end{cases}$$

où f est une fonction lipshitzienne :

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

Le principe de cette méthode est de diviser le domaine X en plusieurs n-rectangles et de construire sur chacun d'eux une fonction borne supérieure constante de la

fonction objectif.
Soit x_i un point dans le n-rectangle X, la fonction borne supérieure de f est définie par : $f_i(x) = f(x_i) + L\|x - x_i\|$, et la fonction borne supérieure constante est le maximum de f_i sur X_i.
L'algorithme divise le domaine X en p n-rectangle $(X_i)_{i=1,\dots,p}$.
Soit $X_i = [a_1^i, b_1^i] \times \ldots \times [a_n^i, b_n^i] = [a^i, b^i]$ un de ces domaines et (c_i) le milieu de (X_i).
La valeur de la fonction borne supérieure constante sur (X_i) est définie par :

$$F_i = f(c_i) + L\frac{\|b^i - a^i\|}{2}$$

On note : $l_j^i = b_j^i - a_j^i \quad , j = 1, \dots, n$

Algorithme

$k \leftarrow 1$;
$x^1 \leftarrow (a_1 + \frac{l_1}{2}, a_2 + \frac{l_2}{2}, \ldots, a_n + \frac{l_n}{2})$;
$D \leftarrow \sqrt{l_1^2 + l_2^2 + \ldots + l_n^2}$;
$x_{opt} \leftarrow x^1$;
$f_{opt} \leftarrow f(x_{opt})$;
$F_{opt} \leftarrow f_{opt} + \frac{1}{2}LD$
$F_1 \leftarrow F_{opt}$;
$P_1 \leftarrow (F_1; (a_1, \ldots, a_n); (l_1, \ldots, l_n))$;
$L \leftarrow \{P_1\}$;
Tant que $F_{opt} - f_{opt} > \epsilon$ *faire*
$P_i = (F_i; (a_1^i, \ldots, a_n^i); (l_1^i, \ldots, l_n^i))$;
$l_{j_0}^i \leftarrow \max_{j=1,2,\ldots,n} l_j^i$;

$$D \leftarrow \left(\sum_{j=1, j \neq j_0}^{n} \left(l_j^i\right)^2 + \left(\frac{l_{j_0}^i}{p}\right)^2 \right)^{\frac{1}{2}};$$

Suprimer P_i de L ;
Pour $k : 1$ à p faire
$A_k \leftarrow (a_1^i, \ldots, a_{j_0-1}^i, a_{j_0}^i + \frac{k-1}{p} l_{j_0}^i, \ldots, a_n^i)$;
$L_k \leftarrow (l_1^i, \ldots, l_{j_0-1}^i, \frac{l_{j_0}^i}{p}, \ldots, l_n i)$;
$F_k \leftarrow f(A_k + \frac{1}{2}L_k) + \frac{1}{2}LD$;
Si $f_{opt} < f(A_k + \frac{1}{2}L_k)$;
$f_{opt} \leftarrow f(A_k + \frac{1}{2}L_k)$;
$x_{opt} \leftarrow A_k + \frac{1}{2}L_k$;
Fin Si ;
Ajouter le sous-problème à L
Fin Pour ;
Eliminer de L tous les sous-problèmes ayant une borne supèrieure inférieure à f_{opt}
Fin Tant que

Remarque. Comme expliqué precedemment, le test d'élimination n'est pas nécessaire pour la convergence de l'algorithme.

2.3.2 Convergence de l'algorithme

On cite ici un théorème de convergence établi par [43].

Théorème 2.1 *L'algorithme de Piyavskii généralisé est convergent, c.â.d il se termine après un nombre fini d'itérations, ou bien on a*

$$\lim_{k\to\infty} F^k_{opt} = \lim_{k\to\infty} f^k_{opt} = f^* = \max_{x\in H} f(x)$$

Démonstration

Soit $\epsilon > 0$

Pour montrer la convergence, on doit trouver une suite $(r_1, \ldots, r_n) \in N^n$ telle que :

$$D = \frac{L}{2}\sqrt{\sum_{i=1}^{n}\left(\frac{l_i}{p^{r_i}}\right)^2} \leq \epsilon$$

$$\frac{LD}{2} \leq \epsilon \quad \Longleftrightarrow \quad D \leq \frac{2\epsilon}{L} \quad \Longleftrightarrow \quad \sum_{i=1}^{n}\left(\frac{l_i}{p^{r_i}}\right)^2 \leq \frac{4\epsilon^2}{L^2}$$

il suffit de choisir :

$$\forall i \quad r_i \geq \log_p\left(\frac{\sqrt{n}Ll_i}{2\epsilon}\right)$$

donc $\exists m \in H \quad \exists k \quad \exists (r_1, \ldots, r_n)$ tel que

$$|F^k_{opt} - f^k(m)| \leq \epsilon$$

où

$$f^k(m) = F^k_{opt} - \frac{LD}{2}$$

Par définieion on a $f^k(m) \leq f^* \leq F^k_{opt}$.

D'où $\exists k \in I\!N \quad |F^k_{opt} - f^*| \leq \epsilon$.

Puisque ϵ est arbitraire, on a :

$$\lim_{k\to\infty} F^k_{opt} = f^*$$

De plus $f^k(m) \leq f^k_{opt} \leq f^*$

ce qui implique que

$$|f^* - f^k_{opt}| \leq \epsilon$$

2.3.3 Optimisation avec contraintes

Soit le problème suivant :

$$\begin{cases} \text{Max} \quad f(x) & x \in X \\ g_i(x) \leq 0 & i = 1, 2, \ldots, k \\ \text{où} & X = [a_1, b_1] \times \ldots \times [a_n, b_n] = [a, b] \end{cases}$$

Supposons que les fonctions contraintes sont lipschitziennes :

$$\forall x, y \in X \times X \quad |g_i(x) - g_i(y)| \leq L_i \|x - y\| \quad i = 1, \ldots, k$$

Pour les problèmes avec contraintes on ajoute deux tests :

Test d'élimination

L'algorithme calcule une borne inférieure pour chaque contrainte dans le n-rectangle X_j :

$$\forall x \in X_j \qquad g_i(x) \geq g_i(c_j) - \frac{L_i D}{2} = LB_{ij}$$

où :

- c_i est le point milieu de X_j
- D est la diagonale X_j

Et l'algorithme élimine les sous-problèmes qui correspondent à une surface non admissible :
Si $LB_{ij} > 0 \quad \forall i$ alors éliminer le sous-problèmes correspondant.

Test de mise à jour

L'algorithme vérifie la nouvelle valeur de x_{opt} et la met à jour si elle satisfait les contraintes.

2.4 Résultats numériques

2.4.1 Problèmes sans contraintes

Numéro de la fonction	Fonction f(x,y)	Intervalle [a,b]	constante de Lipshitz
1	$4x_1x_2\sin(4\pi x_2)$	$[0,1]\times[0,1]$	75
2	$\sin(x_1)\sin(x_1x_2)$	$[0,4]\times[0,4]$	4.299
3	$\frac{\sin(2x_1+x_2)}{\sin(x_2)+2}$	$[-5,5]\times[-5,5]$	2.237
4	$-\prod_{i=1}^{3}(\sum_{k=1}^{5} k\cos((k+1)x_i+k))$	$\prod_{i=1}^{3}[-3,3]$	6080
	$\sin((x_1-1)(x_1-2)(x_2-1))$	$[-1,1]\times[-2,0]$	7.5
6	$-\sin(x_1+x_2)-(x_1-x_2)^2+1.5x_1-2.5x_2-1$	$[-1.5,4]\times[-3,3]$	17.034
7	$-(x_1-2)^2-(x_2-1)^2-\frac{0.04}{-\frac{x_1^2}{4}-x_2^2+1}-\frac{(x_1-2x_2+1)^2}{0.2}$	$[1,2]\times[1,2]$	47.426
8	$-0.1[12+x_1^2]+\frac{(1+x_2^2)}{x_1^2}+\frac{x_1^2x_2^2+100}{x_1^4x_2^4}$	$[1,3]\times[1,3]$	56.852

TABLE 2.1 – Définition des fonctions choisies pour l'évaluation de l'algorithme de Piyavskii sans contraintes.

Les résultats sont stockés dans le tableau 2.2 à la fin du chapitre.

2.4.2 Problèmes avec contraintes

Problème 1

La source de ce problème est [60]. La dimension est $n = 5$ et la fonction objectif est :

$$J(x_1,\ldots,x_5) = 10x_1x_4 - 6x_3x_2^5 + x_2x_1^3 + 9\sin(x_5-x_3) + x_5^4x_4^2 + x_2^3$$

Le problème a 13 contraintes données par :

$$x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 \leq 20$$
$$x_1^2 x_3 + x_4 x_5 \geq -2$$
$$x_2^2 x_4 + 10 x_1 x_5 \geq 5$$
$$-3 \leq x_i \leq 3$$

Problème 2

La source de ce problème est [60]. La dimension est $n = 3$ et la fonction objectif est :

$$J(x_1, x_2, x_3) = (x_1^4 + x_2 + x_3) - (x_1 + x_2^2 - x_3)$$

Le problème a 8 contraintes données par :

$$(x_1 - x_2 - 1.2)^2 + x_2 \leq 4.4$$
$$x_1 + x_2 + x_3 \leq 6.5$$
$$1.4 \leq x_1 \leq 2.0$$
$$1.6 \leq x_2 \leq 2.0$$
$$1.8 \leq x_3 \leq 2.0$$

Problème 3

La source de ce problème est [60]. La dimension est $n = 2$ et la fonction objectif est :

$$J(x_1, x_2) = -(x_1 - 4.2)^2 - (x_2 - 1.9)^2$$

Le problème a 8 contraintes données par :

$$-x_1 + x_2 \leq 3$$
$$x_1 + x_2 \leq 11$$
$$x_1 - x_2 \leq 16$$
$$-x_1 - x_2 \leq -1$$

$$0 \leq x_1 \leq 10$$

$$0 \leq x_2 \leq 5$$

Problème 4

La source de ce problème est [60]. La dimension est $n = 4$ et la fonction objectif est :

$$J(x) = -\{|x_1 + \frac{1}{2}x_2 + \frac{2}{3}x_3 + \frac{3}{4}x_4|^{\frac{3}{2}} + 0.1(x_1 - 0.5x_2 + 0.3x_3 + x_4 - 4.2)^2\}$$

Le problème a 14 contraintes données par :

$$Ax \leq b$$

$$0 \leq x_i \leq 2 \quad i = 1, \ldots, 4$$

Où :

$$A = \begin{pmatrix} 1.2 & 1.4 & 0.4 & 0.8 \\ -0.7 & 0.8 & 0.8 & 0.0 \\ 0.0 & 1.2 & 0.0 & 0.4 \\ 2.8 & -2.1 & 0.5 & 0.0 \\ 0.4 & 2.1 & -1.5 & -0.2 \\ -0.6 & -1.3 & 2.3 & 0.5 \end{pmatrix}, \qquad b = \begin{pmatrix} 6.8 \\ 0.8 \\ 2.1 \\ 1.2 \\ 1.4 \\ 0.8 \end{pmatrix}$$

Dans les tableaux suivants les résultats numériques des problèmes définis ci-dessus. Tous les programmes ont été implémenté en Matlab 6.5.1, sous Windows XP et Pentium (R) 4 CPU 1400 MHZ.

Numéro de la fonction	Valeur optimale trouvée f_{opt}	Précision ϵ	Vecteur optimal trouvé	Nombre d'évaluation	temps (sec)
1	2.5177	10^{-5}	(0.992,0.636)	12901	16.31
2	0.9990	10^{-5}	(1.6049,0.9629)	44275	16.01
3	0.9999	10^{-2}	(4.7142,-1.5729)	19120	16.01
4	2662.3528	0.5	(-0.7736,-0.7736,-1.4320)	28660	26.95
5	0.9999	2×10^{-2}	(-0.1234,-0.3415)	42154	53.51
6	1.9132	6.0×10^{-2}	(-0.5481,-1.5473)	43927	30.50
7	-0.1690	8.0×10^{-2}	(1.7949,1.3779)	57649	70.03
8	-1.7441	0.8	(1.7448,2.0246)	26674	33.67

TABLE 2.2 – Résultats numériques de l'algorithme de Piyavskii sans contraintes.

Numéro de la fonction	Valeur optimale trouvée f_{opt}	Vecteur optimal trouvé	nombre d'évaluation	temps (sec)
1	2411.3513	(-2.2222,2.8888,2.0000,-0.6666,-0.6666)	1855	40.71
2	-4.5955	(1.4012,1.8098,1.8037)	17533	38.2343
3	28.6419	(9.4444,0.8333)	123	27.35
4	6.2331	(1,1,0.7777,1.6666)	2026	40.79

TABLE 2.3 – Résultats numérique de l'algorithme de Piyavskii avec contraintes.

Interprétation des résultats

On remarque que l'algorithme de Piyavskii converge vers l'optimum global, mais en l'implémentant nous avons pu constater qu'il devient lent au voisinage de l'optimum et c'est ce qui explique le temps de calcul élevé par rapport à la dimension des problèmes.

2.5 Conclusion

Dans ce chapitre, on a présenté l'algorithme de Piyavskii qui est un algorithme déterministe établi pour optimiser une fonction Lipshitzienne d'une seule variable sur un intervalle $[a, b]$, puis on a étendu l'algorithme au cas multidimensionel selon l'approche de Shubert [56]. Un théorème de convergence de

l'algorithme a été présenté et des tests numériques sur des problèmes classiques de la litérature ont montré que l'algorithme converge bien vers l'optimum global, mais qu'il est lent au voisinage de l'optimum. En vue d'y remédier, une voie possible est le couplage avec l'algorithme génétique, étudié dans le chapitre 5.

Chapitre 3

Gradient projeté perturbé

3.1 Introduction

Dans ce chapitre, on veut résoudre le problème général suivant :

$$x^* = \text{argmin}\,\{J(x) : x \in C\} \tag{3.1}$$

où $J : I\!R^{n-1} \longrightarrow I\!R$ est une fonction régulière qui n'est pas forcément convexe ou linéaire et C est un ensemble régulier non vide. C est définie par :

$$C = \{x \in I\!R^{n-1} \,/\, h_i(x) \leq 0,\, i = 1, \ldots, p \quad \text{et} \quad h_i(x) = 0,\, i = p+1, \ldots, q\}. \tag{3.2}$$

La fonction J est appelée la fonction objectif, et les fonctions $h_i,\ i = 1, \ldots, q$ sont appelées les fonctions contraintes, et l'ensemble C l'ensemble des contraintes.

Puisque la fonction objectif n'est pas convexe, les méthodes classiques d'optimisation comme le gradient projeté [50], le gradient réduit [57], la méthode SQP [53], ... sont avérées limitées dans leurs applications. Elles ne trouvent que des optimums locaux.
Pour remédier à ce type de problèmes plusieurs approches ont été considérés : algorithme génétique [22], méthode du recuit simulé [51] qui sont des algorithmes stochastiques, la perturbation aléatoire de la méthode du gradient ([9], [58]).

Afin d'aborder plus efficacement le gradient projeté, il est impératif de souligner plus promptement sa perturbation.
En vue d'une élaboration comparative de ces deux méthodes, l'utilisation de fonctions test classiques est significative.

3.2 Notations et hypothèses

Pour rendre le problème 3.1 sous une forme plus simple, on ajoute une variable x_n et une contrainte :

$$h(x) = J(x_1,\ x_2, \ldots,\ x_{n-1}) - x_n \leq 0 \tag{3.3}$$

Le problème 3.1 est donc équivalent au problème suivant :

$$x^* = \text{argmin}\ \{x_n : x \in S\} \tag{3.4}$$

où

$$S = \{x \in I\!R^n \ /\ g_i(x) \leq 0,\ i = 1, \ldots, p+1 \quad \text{et} \quad g_i(x) = 0,\ i = p+2, \ldots, q+1\} \tag{3.5}$$

Les fonctions $g_i :\ i = 1, \ldots, q+1$ sont définies comme suit :

$$\begin{cases} g_i(x) = h_i(x) & \text{pour } i = 1, \ldots, p \\ g_{p+1}(x) = h(x) & h \text{ est définie dans (3.3)} \\ g_i(x) = h_{i-1}(x) & \text{pour } i = p+2, \ldots, q+1 \end{cases}$$

Soit R le réctangle définie par :

$$R = \{x \in I\!R^n \ /\ \underline{r_i} \leq x \leq \overline{r_i}\ ,\ i = 1, \ldots, n\}. \tag{3.6}$$

$\underline{r_i}$ et $\overline{r_i}$ sont choisis de telle manière que tout point de S soit dans R. La longueur maximale de R est donnée par la quantitée L, où

$$L^2 = \sum_{i=1}^{n} (\overline{r_i} - \underline{r_i})^2 = \| \overline{r} - \underline{r} \|^2 \ . \tag{3.7}$$

donc

$$\forall (x_1, x_2) \in R^2 \quad \| x_1 - x_2 \| \leq L. \tag{3.8}$$

La solution du problème 3.4, x^*, est toujours dans la frontière de S notée $fr(S)$ car la fonction objectif est linéaire, $fr(S)$ est définie par :

$$fr(S) = \{\ x \in I\!R^n \ /\ x \in S \text{ et } g_i(x) = 0 \text{ pour au moins un i }\}.$$

La contrainte g_i est active au point x si et seulement si $g_i(x) = 0$. Soit a le nombre des contraintes actives en x. si $g_{i_1}, \ldots, g_{i_a}$ avec $i_1 < \ldots < i_a$ sont les contraintes actives en x, on note $I_a = \{i_1, \ldots, i_a\}$, et

$$g_a(x) = (g_{i_1}(x), \ldots, g_{i_a}(x))^T. \tag{3.9}$$

Soit $A_a(x)$ la matrice jacobienne

$$A_a(x) = \frac{\partial g_a}{\partial x}(x). \tag{3.10}$$

et

$$V_a(x) = (A_a(x)A_a(x)^T)^{-1} \quad , \ x \in \partial S. \tag{3.11}$$

$\forall x \notin \partial S$ on suppose $A_a(x) = 0$, donc

$$\exists \alpha \ / \ \parallel V_a(x) \parallel < \alpha^2 \quad , \ \forall x \in \partial S. \tag{3.12}$$

La matrice $V_a(x), \ x \in \partial S$ est définie positive [50], donc la quantité

$$v_i(x) = \sqrt{V_{aii}(x)} \quad , \ x \in \partial S, \tag{3.13}$$

existe, où $V_{aii}(x)$ sont les élément de la diagonale de V.

On définie la matrice de projection $P_a(x)$ par :

$$P_a(x) = Id - A_a(x)^T V_a(x) A_a(x). \tag{3.14}$$

On a

$$\|P_{a-1} \nabla g_l\| = v_{ll}^{-\frac{1}{2}} \geq \alpha^{-1} \tag{3.15}$$

Soit $C_l(x)$ la matrice hessienne associée à la contrainte $g_l(x), \ l = 1, \ldots, m$

$$(C_l)_{ij} = \frac{\partial^2 h_l}{\partial x_i \partial x_j}. \tag{3.16}$$

On suppose que

$$\exists \gamma \geq 0 \ / \ \parallel C_l(x) \parallel \leq \gamma \quad x \in R. \tag{3.17}$$

On définie une constante λ qui dépend de la géométrie du domaine S par

$$\lambda = \max\{1, \gamma \alpha^2 n^{\frac{3}{2}}\} \geq 1. \tag{3.18}$$

Soit $\theta > \alpha_0$. On note par S_θ l'ensemble

$$S_\theta = \{x \in S \ \mid \ \alpha_0 \leq \mathbf{x}_n < \theta\} \ .$$

3.3 Méthode du gradient projeté

Rosen [50], de par son introduction de la méthode du gradient projeté a ouvert de nouvelles dimensions de résolution.
La méthode de gradient projeté, s'explique par une conjècture :
Si on part d'un point réalisable x_k, on résoud un problème d'interpolation unidimensionnelle pour trouver un pas convenable, on doit faire le long de la direction du vecteur grradient de la fonction objectif, qui n'est d'autre que le vecteur unité constant, pour arriver sur $fr(S)$.
Une fois ce point obtenu, on détermine les contraintes actives en ce point et on projete le vecteur gradient sur le sous espace $TS(x_k)$ tangent à la surface G déterminée par l'intersection des contraintes actives.
Si ce vecteur projection est non nul, son opposé détermine une direction de descente pour l'étape suivante.
Cependant ce vecteur n'est généralement pas une direction réalisable, et pour

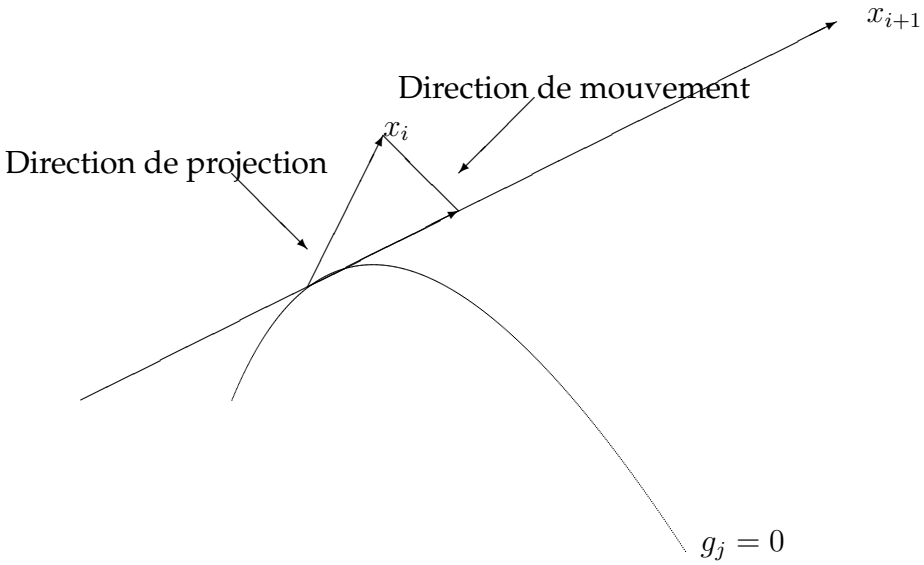

FIGURE 3.1 – Correction du mouvement.

remédier à cette difficulté, on fait un déplacement sequentiel le long de l'opposé du vecteur gradient projeté, puis on corrige le mouvement vers la surface G.

Cette procédure de déplacement et de correction du mouvement introduit un certain nombre de difficultés qui nécessitent une série d'interpolations et solutions d'équations non linéaires.
Pour éviter ce genre de difficulés, une technique itérative est utilisée ; dont l'idée est de corriger le mouvement vers la surface des contraintes saturées en suivant la direction orthogonale au plan tangeant à G en x_k (voir figure 3.1).

3.3.1 Direction de descente

Dans ce qui suit, on note par P_a, A_a et V_a respectivement $P_a(x_k)$, $A_a(x_k)$ et $V_a(x_k)$. Le vecteur gradient de la fonction objectif est :

$$d = (0, \ldots, 0, 1)^T \tag{3.19}$$

On définie$r(x_k) \in I\!R^m$ par :

$$r(x_k) = V_a A_a d. \tag{3.20}$$

et

$$\beta_i(x_k) = \frac{r_i(x_k)}{2v_i(x_k)} \quad , \; i = 1, \ldots, m. \tag{3.21}$$

si l est l'index correspondant à la valeur maximale de $\beta_i\,(x_k)$, on note par β_k cette valeur maximale :

$$\beta_k = \beta_l\,(x_k) = \max\left\{\beta_i\,(x_k) \;,\; i = 1, \ldots, m\right\} \tag{3.22}$$

et

$$b_k = \frac{1}{2}\max\left\{\|P_a \mathbf{d}\|\,,\; \beta_k\right\}\;. \tag{3.23}$$

b_k vérifie [9]

$$b_k \leq 1 \quad , \quad \forall x_k \in \partial S \quad . \tag{3.24}$$

on définie :

$$I_{a-1} = \{i_1, \ldots, i_a\} - \{l\}.$$

g_{a-1}, A_{a-1}, V_{a-1} et P_{a-1} sont définies de la même manière que dans 3.9, 3.10, 3.11 et 3.14.
La direction de déscente d est déterminée en utilisant le résultat suivant [9, 8] :

Proposition 3.1 *Soit $x_k \in \partial S$ vérifiant*

$$\|P_a \mathbf{d}\| < \beta_k = 2b_k \ ,$$

donc

$$\|P_{a-1}\left(\mathbf{x}_k\right)\mathbf{d}\| \geq \frac{r_\ell\left(\mathbf{x}_k\right)}{v_i\left(\mathbf{x}_k\right)} - \|P_a \mathbf{d}\| \geq \beta_k = 2b_k \ . \qquad \triangle$$

Démonstration

Les vecteurs $\nabla g_i = \nabla g_i(x),\ i = 1, \ldots, q$ engendrent le sous espace vectoriel $TS^{\perp}(x)$, donc

$$\mathbf{d} = P_a(x)\mathbf{d} + \sum_{i=1}^{a} r_i \nabla g_i = P_a(x)\mathbf{d} + A_q^T(x) r \tag{3.25}$$

Multiplions les deux membres de l'équation 3.25 par $V_a(x)A_a(x)$. Puisque TS est contenu dans le sous espace correspondant à P_{a-1}, on a

$$P_{a-1}P_a\mathbf{d} = \mathbf{d}$$

et

$$P_{a-1}\mathbf{d} = P_a\mathbf{d} + r_l P_{a-1}\nabla g_l$$

Les relations 3.15 et 3.20 impliquent

$$\|P_{a-1}\mathbf{d}\| \geq r_l v_{ll}^{-\frac{1}{2}} - \|P_a\mathbf{d}\| \geq \beta_k = 2b_k. \qquad \triangle$$

La proposition implique que

$$\|\mathbf{d}_k\| \geq 2b_k \quad . \tag{3.26}$$

Donc la direction de déscente est définie par :

$$\mathbf{d}_k = \ - P_k\mathbf{d} \tag{3.27}$$

avec

$$P_k = P_a \ , \ \text{si} \ \ \|P_a\mathbf{d}\| \geq \beta_k \quad ; \quad P_k = P_{a-1} \ , \ \text{sinon} \ . \tag{3.28}$$

3.3.2 Correction vers la frontière

Une fois la direction de descente détérminée, un nouveau point xt_k est déterminé :

$$xt_k = x_k + \eta_k \mathbf{d}_k \ .$$

où

$$\frac{8\alpha\delta d_k}{b_k} \leq \eta_k \leq \frac{\alpha d_k b_k}{6\lambda} \ \ ; \ \ \mathbf{d}_k = \|d_k\| \quad . \tag{3.29}$$

Si xt_k est un point intérieur :

$$x_{k+1} = x_k + \xi_k \mathbf{d}_k \ \ ; \ \ \xi_k = Arg \ \min \{t > \eta_k \ \ | \ \ x_k + t\mathbf{d}_k \notin S\} \, .$$

Si xt_k est un point irréalisable, la correction est faite en utilisant une méthode itérative : on génère une suite de points $\left\{x^{(j)}\right\}_{j\geq 0}$, tel que $x^{(0)} = xt_k$ et

$$x^{(j+1)} = x^{(j)} - \varphi_k\left(x^{(j)}\right) \quad , \quad \varphi_k\left(x^{(j)}\right) = A_k^T V_k g_k\left(x^{(j)}\right) \ \ ; \tag{3.30}$$

où

$$A_k = A_a \ , \ V_k = V_a \ , \ g_k\left(x^{(j)}\right) = g_a\left(x^{(j)}\right) \ \ , \ \text{si} \ \ \|P_k\mathbf{d}\| \geq \beta_k \quad ;$$

$$A_k = A_{a-1} \ , \ V_k = V_{a-1} \ , \ g_k\left(x^{(j)}\right) = g_{a-1}\left(x^{(j)}\right) \ \ , \ \text{si} \ \ \|P_k\mathbf{d}\| < \beta_k \quad .$$

Soit $x_k \in E$ et $\delta > 0$ verifiant

$$\|g_a\left(x_k\right)\| \leq \delta \leq \frac{b_k^2}{48\lambda} \tag{3.31}$$

Soit j satisfaisant

$$j \geq 1.443 \ \log\left(24\lambda\delta\right)^{-1} \ . \tag{3.32}$$

On a les théorèmes suivant qui assurent la convergence vers un point de la frontière (pour la preuve de ces théorèmes voir [50, 8]) :

Théorème 3.1 *supposons que les équations 3.31 et 3.32 sont satisfaites et :* $2b_k = \|P_k\mathbf{d}\| \geq \beta_k$. *Donc*

$$\left\|g_a\left(x^{(j)}\right)\right\| \leq \delta \ . \tag{3.33}$$

De plus,

$$J\left(x_k\right) - J\left(x^{(j)}\right) \geq \frac{3}{2}\eta_k b_k \geq 12\alpha\delta \ . \qquad \triangle \tag{3.34}$$

Théorème 3.2 *supposons que les équations 3.31 et 3.32 sont satisfaites et :* $2b_k = \beta_k > \|P_k\mathbf{d}\|$. *Donc*

$$\left\|g_{a-1}\left(x^{(j)}\right)\right\| \leq \delta \tag{3.35}$$

et

$$J\left(x_k\right) - J\left(x^{(j)}\right) \geq \frac{3}{2}\eta_k b_k \geq 12\alpha\delta\ . \tag{3.36}$$

Si, en plus, $\beta_k \leq (4\alpha)^{-1}$, *alors*

$$g_l\left(x^{(j)}\right) \leq -\frac{\eta_k b_k}{2} \leq -4\alpha\delta\ . \qquad \triangle \tag{3.37}$$

Les théorèmes 3.1 et 3.2 montrent que, pour $x_k \in fr(S)$, la suite $\{x^{(j)}\}_{j\geq 0}$ converge vers un point $x_{k+1} \in fr(S)$.

3.4 Perturbation aléatoire du gradient projeté

L'algorithme du gradient projeté n'est pas efficace quand la fonction objectif J n'est pas convexe car il trouve seulement des minimums locaux au su et au vu des algorithmes déterministes.
Des auteurs [9] ont suggéré de remplacer la suite engendrée par l'algorithme du gradient projeté , $\{x_k\}_{k\geq 0}$, par une nouvelle suite aléatoire $\{X_k\}_{k\geq 0}$, en utilisant une suite aléatoire $(Z_k)_{k\geq 0}$, où Z_k est généré en utilisant la matrice de projection : si Z est un vecteur gaussien, donc $Z_k = P_k Z$, la réciprocité est acquise.

Soit $\{\xi_k\}_{k\geq 0}$ une suite décroissante de nombres strictement positifs, convergeant vers 0, ϕ la densité de probabilité de Z_k, et Φ_k sa fonction cumulative. On suppose qu'il existe une fonction *décroissante* $\psi\ :\ I\!R \longrightarrow I\!R$, indépendante de k, telle que

$$\forall\ z \in TS\left(X_k\right)\ :\ \phi(z) \geq \psi\left(\|z\|\right) > 0 \tag{3.38}$$

La direction de descente perturbée et le nouveau point obtenu par le gradient projeté perturbé seront respectivement :

$$\mathbf{D}_k = \mathbf{d}_k + \xi_k Z_k\ .$$

$$XT_k = X_k + \eta_k \mathbf{D}_k\ .$$

où η_k est choisi en utilisant l'equation 3.29.

XT_{k+1} doit être corrigé de la même manière que dans la section (3.3.2). Pour des valeurs de $XT_k \notin S$, X_{k+1} est généré en utilisant une méthode itérative, comme suit

$$X^{(0)} = XT_k \; ; \; X^{(j+1)} = X^{(j)} - \varphi_k \left(X^{(j)} \right) \quad . \tag{3.39}$$

Ces itérations sont stopées après un nombre fini d'itérations $j > 0$ fourni par les théorèmes 3.1 et 3.2, et on pose $X_{k+1} = X^{(j+1)}$.

Soit $\varepsilon > 0$ donné. Les théorèmes 3.1 et 3.2 montrent l'existence d'un index $H(\delta)$ tel que

$$\left\| g_a \left(X^{(j)} \right) \right\| \leq \varepsilon \;\; \forall \, j \geq H \left(\varepsilon \right) \; .$$

Si les itérations 3.39 sont stoppées pour une valeur $j \geq H(\varepsilon) - 1$, donc

$$X^{(j+1)} = X_k - \sum_{i=0}^{j} A_k^T V_k g_k \left(X^{(j)} \right) + \eta_k d_k \; .$$

Alors, le gradient projeté perturbé génére une suite de vecteurs aléatoires telle que

$$\begin{aligned} X_{k+1} &= X_k - \sum_{i=0}^{j} A_k^T V_k g_k \left(X^{(i)} \right) + \eta_k d_k + \eta_k \xi_k Z \\ &= Q_k \left(X_k \right) + \eta_k \xi_k Z \end{aligned}$$

où

$$Q_k \left(X_k \right) = X_k - \sum_{i=0}^{j} A_k^T V_k g_a \left(X^{(i)} \right) + \eta_k \mathbf{d}_k \; .$$

On a

$$\begin{aligned} P \left(X_{k+1} < y \mid X_k = x \right) &= P \left(Q_k \left(X_k \right) + \eta_k \xi_k Z < y \mid X_k = x \right) \\ &= P \left(Z < \frac{y - Q_k \left(x \right)}{\eta_k \xi_k} \right) \; . \end{aligned}$$

Donc, la fonction cumulative de X_{k+1} est

$$F_{k+1} \left(y \mid X_k = x \right) = \Phi \left(\frac{y - Q_k \left(x \right)}{\eta_k \xi_k} \right) \; .$$

et sa densité de probabilité f_{k+1} est

$$f_{k+1}\left(y \mid X_k = x\right) = \frac{1}{\eta_k^n \xi_k^{m_k}} \phi\left(\frac{y - Q_k\left(x\right)}{\eta_k \xi_k}\right) . \tag{3.40}$$

On obtient le résultat suivant

Proposition 3.2 *Soit $x \in S$ et $y \in S_\theta$,*

$$Q_a\left(x\right) = x - \sum_{i=0}^{j} A_a\left(x\right)^T V_a\left(x\right) g_a\left(x^{(i)}\right) - \eta_k P_a d \quad .$$

Alors :

$$\left\| y - Q_a\left(x\right) \right\| \leq M$$

où

$$M = 2L + \frac{\sqrt{n}\Lambda\alpha^2}{12\lambda} + \frac{\alpha}{6\lambda} .$$

Démonstration

Soit $\theta > \alpha_0$. $S_\theta \subset S$ et on a, selon l'équation 3.8, $\|y\| \leq L, \forall x \in S_\theta$.

Soit $x \in S$. Donc (voir [12])

$$\left\|A_a^T\right\| \leq \left(\sum_{i,j} A_{a_{ij}}^2\right)^{1/2} = \left(\sum_{i=0}^{m} \left\|\nabla h_i\right\|^2\right)^{\frac{1}{2}} ,$$

Puisque les contraintes sont régulières il existe un nombre Λ tel que

$$\Lambda \geq \max\left\{ \left\|\nabla g_i\left(x\right)\right\| \ : \ x \in S_R \ ; \ 1 \leq i \leq m \right\} . \tag{3.41}$$

et donc

$$\left\|A_a^T\right\| \leq \sqrt{m}\Lambda \leq \sqrt{n}\Lambda. \tag{3.42}$$

Soit $x \in S$ et $y \in S_\theta$. On a :

$$\begin{aligned} \left\| y - Q_a\left(x\right) \right\| &= \left\| y - x + \sum_{i=0}^{j} A_a^T V_a g_a\left(x^{(i)}\right) + \eta_k P_a d \right\| \\ &\leq \|y\| + \|x\| + \sum_{i=0}^{j} \left\|A_a^T\right\| \left\|V_a\right\| \left\|g_a\left(x^{(i)}\right)\right\| + \eta_k. \end{aligned}$$

On a (voir [50]),

$$\left\|A_a^T V_a w\right\| \leq \alpha \left\|w\right\| \quad , \quad \forall w \in E \quad .$$

De cette inégalité, et les équations 3.24 and 3.29, on a

$$\|y - Q_a(x)\| \leq 2L + \sum_{i=0}^{j} \left\|A_a^T\right\| \|V_a\| \left\|g_a\left(x^{(i)}\right)\right\| + \frac{\alpha}{6\lambda}.$$

En utilisant les équations 3.12, 3.42 et le fait que $\left\|g_a\left(x^{(j)}\right)\right\| \leq \left(\frac{1}{2}\right)^j (24\lambda)^{-1}$ (voir [8]), on a

$$\left\|A_a(x)^j\right\| \|V_a(x)\| \left\|g_a\left(x^{(j)}\right)\right\| \leq \sqrt{n}\Lambda\alpha^2\left(\frac{1}{2}\right)^j (24\lambda)^{-1}.$$

Donc

$$\begin{aligned}
\|y - Q_a(x)\| &\leq 2L + \sum_{i=0}^{j} \sqrt{n}\Lambda\alpha^2\left(\frac{1}{2}\right)^i (24\lambda)^{-1} + \frac{\alpha}{6\lambda} \\
&\leq 2L + \sqrt{n}\Lambda\alpha^2(24\lambda)^{-1} \sum_{i=0}^{\infty}\left(\frac{1}{2}\right)^i + \frac{\alpha}{6\lambda} \\
&\leq 2L + \frac{\sqrt{n}\Lambda\alpha^2}{12\lambda} + \frac{\alpha}{6\lambda} \\
&= M. \qquad \triangle
\end{aligned}$$

Cette procédure génére une suite $U_k = J(X_k)$. Par construction, cette suite est décroissante et bornée inférieurement par α_0. Donc, il exixte $U \geq \alpha_0$ tel que $U_k \longrightarrow U$ pour $k \longrightarrow +\infty$. La convergence vers un minimum global est assurée par le résultat suivant (voir [44], [42]) :

Lemme 3.1 *Soit* $\{U_n\}_{n\geq 0}$ *une suite* décroissante, bornée inférieurement par α_0. *Donc, il existe* U *tel que*

$$U_n \longrightarrow U \quad \text{for } n \to +\infty .$$

Supposons en plus que pour chaque $\theta \in]\alpha_0, \alpha_1]$, *on peut trouver une suite de nombres strictement positifs* $\{c_n(\theta)\}_{n\geq 0}$ *tel que*

$$\forall\, n \geq 0 \;:\; P(U_{n+1} < \theta \mid U_n \geq \theta) \geq c_n(\theta) > 0\,;\; \sum_{n=0}^{+\infty} c_n(\theta) = +\infty. \tag{3.43}$$

Donc $U = \alpha_0$ *presque sûrement.*

Le lemme 3.1 est appliqué à la suite $\{U_n\}_{n\geq 0}$ comme suit :

Théorème 3.3 *Supposons que $X_0 \in S$; 3.38 est satisfaite et*

$$\sum_{k=0}^{+\infty} \frac{1}{\xi_k} = +\infty \tag{3.44}$$

Donc

$$U = \alpha_0 \text{ presque sûrement}.$$

Démonstration
On a

$$P\left(U_{k+1} < \theta \mid U_k \geq \theta\right) = P\left(X_{k+1} \in S_\theta \mid X_k \notin S_\theta\right) . \tag{3.45}$$

Dés que $\{X_k\}_{k\geq 0} \subset S$, on a aussi (voir [9]) :

$$P(X_{k+1} \in S_\theta, X_k \notin S_\theta) = \int_{S-S_\theta} P(X_k \in dx) \int_{S_\theta} f_{k+1}(y \mid X_k = x)dy . \tag{3.46}$$

Puisque

$$P(X_{k+1} \in S_\theta \mid X_k \notin S_\theta) = \frac{P(X_{k+1} \in S_\theta, X_k \notin S_\theta)}{P(X_k \notin S_\theta)} ,$$

on a

$$P\left(X_{k+1} \in S_\theta \mid X_k \notin S_\theta\right) \geq \inf_{x\in S-S_\theta} \left\{ \int_{S_\theta} f_{k+1}(y \mid X_k = x)dy \right\} ,$$

De l'équation 3.40, on a

$$\begin{aligned} P\left(X_{k+1} \in S_\theta \mid X_k \notin S_\theta\right) &\geq \frac{1}{\eta_k^n \xi_k^n} \inf_{x\in S-S_\theta} \left\{ \int_{S_\theta} \phi_k \left(\frac{y - Q_k\left(x\right)}{\eta_k \xi_k} \right) dy \right\} \\ &\geq \frac{\text{meas}(S_\theta)}{\eta_k^n \xi_k^n} \inf_{x\in S-S_\theta, y\in S_\theta} \left\{ \phi_k \left(\frac{y - Q_k\left(x\right)}{\eta_k \xi_k} \right) \right\} . \end{aligned}$$

En utilisant 3.29, 3.38 et la proposition 3.2, on a

$$P\left(X_{k+1} \in S_\theta \mid X_k \notin S_\theta\right) \geq \frac{\text{meas}(S_\theta)}{\xi_k^n} \left(\frac{6\lambda}{\alpha} \right)^n \psi \left(\frac{M}{\eta_k \xi_k} \right) .$$

La proposition (3.1) implique que

$$\psi \left(\frac{M}{\eta_k \xi_k} \right) \geq \psi \left(\frac{M}{16\alpha\delta\xi_k} \right) \geq \psi \left(\frac{M}{16\alpha\delta\xi_0} \right) .$$

Alors,

$$P\left(X_{k+1} \in S_\theta \mid X_k \notin S_\theta\right) \geq c_\theta\left(k\right) \tag{3.47}$$

avec

$$c_\theta\left(k\right) = \frac{\gamma\left(\theta\right)}{\xi_k^n} \; ; \; \gamma\left(\theta\right) = \text{meas}(S_\theta) \left(\frac{6\lambda}{\alpha}\right)^n \psi\left(\frac{M}{16\alpha\delta\xi_0}\right) \; .$$

Puisque J est continue alors :

$$\forall \theta \in \left]\alpha_0, \alpha_1\right[\; : \; \text{meas}\left(S_\theta\right) > 0 \; . \tag{3.48}$$

et alors

$$\forall \theta \in \left]\alpha_0, \; \alpha_1\right[: c_\theta\left(k\right) > 0 \; .$$

selon 3.44 on a

$$\sum_{k=0}^{+\infty} c_\theta\left(k\right) = \; +\infty \; .$$

et en appliquant le lemme 3.1 : $U = \alpha_0$ presque surêment. $\triangle$

3.5 Résultats numériques

Dans cette section, on donnera des résultats numériques de 6 exemples académiques implémentés en utilisant le gradient projeté et le gradient projeté perturbé, pour comparer les deux algorithmes.

3.5.1 Problèmes

Problème 1

La source de ce problème est [15]. La dimension est $n = 8$ et la fonction objectif est :

$$J(x) = x_1x_2x_3 + x_1x_4x_5 + x_2x_4x_6 + x_6x_7x_8 + x_2x_5x_7$$

Le problème a 26 contraintes données par :
$2x_1 + 2x_4 + 8x_8 \geq 12$; $\quad 11x_1 + 7x_4 + 13x_6 \geq 41$; $\quad 6x_2 + 9x_4x_6 + 5x_7 \geq 60$;
$3x_2 + 5x_5 + 7x_8 \geq 42$; $\quad 6x_2x_7 + 9x_3 + 5x_5 \geq 53$; $\quad 4x_3x_7 + x_5 \geq 13$;
$2x_1 + 4x_2 + 7x_4 + 3x_5 + x_7 \leq 69$; $9x_1x_8 + 6x_3x_5 + 4x_3x_7 \leq 47$;
$12x_2 + 8x_2x_8 + 2x_3x_6 \leq 73$; $\quad x_3 + 4x_5 + 2x_6 + 9x_8 \leq 31$;

$0 \leq x_i \leq 7\,, \quad i = 1,\ 3,\ 4,\ 6,\ 8\,; \quad 0 \leq x_i \leq 15\,, \quad i = 2,\ 5,\ 7\,.$

Dans la source la valeur optimale de J est :
$J(x^*) = 68.1779$ et $x^* = (0.91,\ 2.19,\ 0.96,\ 5.08,\ 7.08,\ 0.85,\ 1.53,\ 0)$

Problème 2

La source de ce problème est [49]. La dimension est $n = 5$ et la fonction objectif est :

$$J(x) = 5.3578x_3^2 + 0.8357x_1x_5 + 37.2392x_1$$

Le problème a 16 contraintes données par :
$0.00002584x_3x_5 - 0.00006663x_2x_5 - 0.0000734x_1x_4 \leq 1$;
$0.00085307x_2x_5 + 0.00009395x_1x_4 - 0.00033085x_3x_5 \leq 1$;
$1330.3294x_2^{-1}x_5^{-1} - 0.42x_1x_5^{-1} - 0.30586x_2^{-1}x_3^2x_5^{-1} \leq 1$.
$0.00024186x_2x_5 + 0.00010159x_1x_2 + 0.00007379x_3^2 \leq 1$;
$2275.1327x_3^{-1}x_5^{-1} - 0.2668x_1x_5^{-1} - 0.40584x_4x_5^{-1} \leq 1$.
$0.00029955x_3x_5 + 0.00007992x_1x_3 - 0.00012157x_3x_4 \leq 1$;
$78 \leq x_1 \leq 102$;
$33 \leq x_2 \leq 45$;
$27 \leq x_3 \leq 45$;
$27 \leq x_4 \leq 45$;
$27 \leq x_5 \leq 45$;

Dans la source la valeur optimale de J est :
$J(x^*) = 10127.13$ et $x^* = (78,\ 33,\ 29.998,\ 45,\ 36.7673)$

Problème 3

La source de ce problème est [49]. La dimension est $n = 4$ et la fonction objectif est :

$$J(x) = 168x_1x_2 + 3651.2x_1x_2x_3^{-1} + 40000x_4^{-1}$$

Le problème a 11 contraintes données par :
$1.0425x_1x_2^{-1} \leq 1$;
$0.00035x_1x_2 \leq 1$;

$1.25x_1^{-1}x_4 + 41.63x_1^{-1} \leq 1$;
$40 \leq x_1 \leq 44$;
$40 \leq x_2 \leq 45$;
$60 \leq x_3 \leq 70$;
$0.1 \leq x_4 \leq 1.4$;

Dans la source la valeur optimale de J est :
$J(x^*) = 623370.0754$ et $x^* = (43.02,\ 44.85,\ 66.39,\ 1.11)$

Problème 4

La source de ce problème est [59]. La dimension est $n = 2$ et la fonction objectif est :

$$J(x) = 2x_1^2 - 1.05x_1^4 + \frac{1}{6}x_1^6 - x_1x_2 + x_2^2$$

Le problème a 4 contraintes données par :
$-3 \leq x_1 \leq 3$;
$-3 \leq x_2 \leq 3$;

Théoriquement la valeur optimale de J est :
$J(x^*) = 0$ and $x^* = (0,\ 0)$

Problème 5

La source de ce problème est [41]. La dimension est $n = 3$ et la fonction objectif est :

$$J(x) = -4x_2 + (x_1 - 1)^2 + x_2^2 - 10x_3^2$$

Le problème a 13 contraintes données par :
$x_1^2 + x_2^2 + x_3^2 \leq 2; \qquad -\sqrt{2} \leq x_2 \leq \sqrt{2};$
$(x_1 - 2)^2 + x_2^2 + x_3^2 \leq 2; \qquad -\sqrt{2} \leq x_3 \leq \sqrt{2};$
$2 - \sqrt{2} \leq x_1 \leq \sqrt{2};$

Dans la source la valeur optimale de J est :
$J(x^*) = -13.5661$ et $x^* = (0.9971,\ 0.1818,\ 0.9803)$

Problème 6

La source de ce problème est [59]. La dimension est $n = 2$ et la fonction objectif est :

$$J(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 - x_1x_2 - 4x_2^2 + 4x_2^4$$

Le problème a 4 contraintes données par :
$-3 \leq x_1 \leq 3$;
$-3 \leq x_2 \leq 3$;

Théoriquement la solution de J est :
$J(x^*) = -1.03162844$ et $x^* = (0.08983714,\ 0.71269875)$

3.5.2 Résultats numériques

Dans les tableaux suivants les résultats numériques des problèmes définies dans la section (3.5.1), en utilisant le gradient projeté et le gradient projeté perturbé, PN est le numéro du problème, x^* le vecteur optimal et $J(x^*)$ la valeur optimale de la fonction objectif.
Tous les programmes ont été implémentés en Matlab 6.5.1, sous Windows XP et Pentium (R) 4 CPU 1400 MHZ.

PN	x^*	J(x^*)	temps(sec)
1	$(5,\ 5,\ 1,\ 2,\ 6,\ 3,\ 2,\ 0)$	175	0.641
2	$(78,\ 33,\ 30,\ 31,\ 42)$	10484	0.062
3	$(40,\ 40,\ 60,\ 0.1)$	766170	9.64
4	$(-2.2863,\ -1.3694)$	4.3130	0.031
5	$(1,\ 0.7090,\ -0.7052)$	-7.3064	0.5
6	$(-2,\ -1)$	1.7333	0.01

TABLE 3.1 – Résultats numériques du gradient projeté

PN	x^*	J(x^*)	temps(sec)
1	$(0,\ 2.3747,\ 1.0051,\ 5.3818,\ 6.7588,\ 0.7843,\ 1.5524,\ 0.1546)$	35.1285	5.3440
2	$(78,\ 33,\ 29.997,\ 45,\ 36.772)$	10123	13
3	$(40,\ 40,\ 70,\ 0.5)$	538510	19.22
4	$(-0.0555,\ -0.8379)$	0.6617	0.781
5	$(2-\sqrt{2},\ -\sqrt{2},\ -\sqrt{2})$	-12.1716	1.03
6	$(-1.7029,\ -0.7914)$	-0.2152	0.18

TABLE 3.2 – Résultats numériques du gradient projeté perturbé

Interprétation des résultats

L'algorithme du gradient projeté perturbé donne de bons résultats comparé au gradient projeté, mais ne converge pas toujours vers l'optimum global du problème quand le choix du point initial n'est pas adéquat.

3.6 Conclusion

Dans ce chapitre, nous avons étudié le comportement du gradient projeté dans les situations non convexes. Les résultats obtenus mettent en évidence des limitations de l'approche choisie. Il semble nécessaire d'envisager une autre méthode que celle du gradient projeté, ce qui sera abordé dans les prochains chapitres.

Chapitre 4

Méthode SQP perturbée

4.1 Introduction

La méthode séquentielle de programmation quadratique abréviée par SQP [53] (cette nomenclature est due à son nom en anglais *sequential quadratic programming*) est une méthode d'optimisation qui s'est avéré beaucoup plus performante que celles basées sur le gradient.
Pour plus d'informations sur ladite méthode, on vous invite à vous référer à ([18], [47]).
A chaque itération, une approximation du hessien du lagrangien du problème est faite en utilisant une méthode quasi-newtonienne, ce qui va permettre de générer un sous-problème QP dont la solution va être utilisée pour former la direction de recherche.
Dans ce chapitre, on étudie deux algorithmes SQP pour les problèmes non linéaires, l'un pour les contraintes de type égalité et l'autre de type inégalité.
Dans la section (4.4), on modifie ces algorithmes en introduisant un pas de recherche linéaire et en mettant à jour la formule de Broyden-Fletcher-Goldfarb-Shanno (BFGS) utilisée pour l'estimation du hessien du Lagrangien.
Néanmoins, la méthode SQP ne donne pas l'optimum global dans le cas où la fonction objectif n'est pas convexe. A l'image du chapitre précédent, la perturbation nous permet de l'améliorer et de l'assujétir à diverses contraintes.
Des résultats numériques sont implémentés pour montrer l'efficacité de la méthode SQP perturbée.

4.2 SQP pour les problèmes avec contraintes de type égalité

Considérons le problème

$$x^* = \text{argmin}\,\{f(x) : x \in C\} \tag{4.1}$$

$$C = \{x \in I\!R^n \,/\, a_i(x) = 0,\, i = 1, \ldots, p\}. \tag{4.2}$$

où $f : I\!R^n \longrightarrow I\!R$ et $a_i : I\!R^n \longrightarrow I\!R,\ i = 1, \ldots, p$ sont des fonctions régulières continues et leurs dérivées secondes sont aussi continues. On sait que la condition nécessaire du premier ordre pour que x^* soit un optimum local du problème 4.1, est qu'il existe un $\lambda^* \in I\!R^p$ tel que

$$\nabla L(x^*, \lambda^*) = 0 \tag{4.3}$$

où L est le Lagrangien définie par

$$L(x, \lambda) = f(x) - \sum_{i=1}^{p} \lambda_i a_i(x)$$

Si l'ensemble $\{x_k, \lambda_k\}$ est l'itération k, on a besoin de trouver un couple $\{\delta_x, \delta_\lambda\}$ tel que l'itération suivante $\{x_{k+1}, \lambda_{k+1}\} = \{x_k + \delta_x, \lambda_k + \delta_\lambda\}$ soit plus proche de l'optimum $\{x^*, \lambda^*\}$. Si on approche $\nabla L(x_{k+1}, \lambda_{k+1})$ en utilisant les deux premiers termes du développement de Taylor de ∇L en $\{x_k, \lambda_k\}$, i.e,

$$\nabla L(x_{k+1}, \lambda_{k+1}) \approx \nabla L(x_k, \lambda_k) + \nabla^2 L(x_k, \lambda_k) \begin{bmatrix} \delta_x \\ \delta_\lambda \end{bmatrix}$$

donc $\{x_{k+1}, \lambda_{k+1}\}$ est l'approximation de $\{x^*, \lambda^*\}$ si le couple $\{\delta_x, \delta_\lambda\}$ satisfait la condition

$$\nabla^2 L(x_k, \lambda_k) \begin{bmatrix} \delta_x \\ \delta_\lambda \end{bmatrix} = -\nabla L(x_k, \lambda_k) \tag{4.4}$$

On peut écrire l'équation 4.4 en fonction du Hessien du Lagrangien, **W**, pour $\{x, \lambda\} = \{x_k, \lambda_k\}$ et le Jacobien, **A**, pour $x = x_k$, comme suit

$$\begin{bmatrix} \mathbf{W}_k & -\mathbf{A}_k^T \\ \mathbf{A}_k & 0 \end{bmatrix} \begin{bmatrix} \delta_x \\ \delta_\lambda \end{bmatrix} = \begin{bmatrix} -\mathbf{A}_k^T \lambda_k - g_k \\ a_k \end{bmatrix} \tag{4.5}$$

où

$$\mathbf{W}_k = \nabla_x^2 f(x_k) - \sum_{i=1}^{p} (\lambda_k)_i \nabla_x^2 a_i(x_k) \tag{4.6}$$

$$\mathbf{A}_k = \begin{bmatrix} \nabla_x^T a_1(x_k) \\ \nabla_x^T a_2(x_k) \\ . \\ . \\ . \\ \nabla_x^T a_p(x_k) \end{bmatrix} \tag{4.7}$$

$$g_k = \nabla_x f(x_k) \tag{4.8}$$

$$a_k = [a_1(x_k)\ a_2(x_k)\ \ldots\ a_p(x_k)]^T \tag{4.9}$$

Si $\mathbf{W}_k$ est définie positive et $\mathbf{A}_k$ a un rang non nul, alors la matrice à gauche de l'équation 4.5 est non singulière et symétrique et le système d'équations en 4.5 peut être résolu facilement.
L'équation 4.5 peut s'écrire aussi sous la forme

$$\mathbf{W}_k \delta_x + g_k = \mathbf{A}_k^T \lambda_{k+1} \tag{4.10}$$

$$\mathbf{A}_k \delta_x = -a_k \tag{4.11}$$

Ces équations peuvent être interprétées comme les conditions nécessaires du premier ordre pour que δ_x soit un minimum local du problème QP

$$\begin{array}{l} \text{minimiser } \frac{1}{2}\delta^T \mathbf{W}_k \delta + \delta^T g_k \\ \text{sous la contrainte : } \mathbf{A}_k \delta = -a_k \end{array} \tag{4.12}$$

Une fois le minimum, δ_x, est obtenu, on pose $x_{k+1} = x_k + \delta_x$ et le vecteur Lagrangien est obtenu en utilisant l'équation 4.10

$$\lambda_{k+1} = (\mathbf{A}_k \mathbf{A}_k^T)^{-1} \mathbf{A}_k (\mathbf{W}_k \delta_x + g_k) \tag{4.13}$$

Si x_{k+1} et λ_{k+1} sont connus, les quantités $\mathbf{W}_{k+1}$, g_{k+1}, $\mathbf{A}_{k+1}$ et a_{k+1} peuvent être évaluées. Les itérations continuent jusqu'à ce que $\|\delta_x\|$ soit suffisamment petit pour terminer l'algorithme.
La méthode SQP pour les problèmes de type égalité peut être implémenté par l'algorithme suivant.

Algorithme

1. Poser $\{x, \lambda\} = \{x_0, \lambda_0\}$, $k = 0$ et initialiser une tolérence ϵ.
2. A l'itération k, évaluer W_k, A_k, g_k et a_k en utilisant les relations 4.6, 4.7, 4.8 et 4.9.
3. Résoudre le problème QP dans l'équation 4.12 pour δ et calculer le multiplicateur de Lagrange λ_{k+1} en utilisant la relation 4.13.
4. Poser $x_{k+1} = x_k + \delta_x$. Si $\|\delta_x\| \leq \epsilon$ alors $x^* = x_{k+1}$; sinon, poser $k = k + 1$ et aller à l'étape 2.

4.3 SQP pour les problèmes avec contraintes de type inégalité

Dans cette section, on considère le problème suivant

$$x^* = \text{argmin}\,\{f(x) : x \in C\} \tag{4.14}$$

$$C = \{x \in I\!R^n \,/\, c_i(x) \geq 0,\, i = 1, \ldots, q\}. \tag{4.15}$$

où $f : I\!R^n \longrightarrow I\!R$ et $c_i : I\!R^n \longrightarrow I\!R,\ i = 1, \ldots, q$ sont des fonctions régulières continues et leurs dérivées secondes sont aussi continues. Dans cette section, on veut trouver un couple $\{\delta_x,\, \delta_\mu\}$ pour l'itération k $\{x_k,\, \mu_k\}$ tel que l'itération suivante $\{x_{k+1},\, \mu_{k+1} = \{x_k + \delta_x,\, \mu_k + \delta_\mu\}\}$ approche les conditions de Karush-Kuhn-Tucker (KKT)

$$\nabla_x L(x,\, \mu) = 0$$

$$c_j(x) \geq 0 \quad \text{pour } j = 1,\, 2, \ldots,\, q$$

$$\mu \geq 0$$

$$\mu_j c_j(x) = 0 \quad \text{pour } j = 1,\, 2, \ldots,\, q$$

On a

$$\nabla_x L(x_{k+1},\, \mu_{k+1}) \approx \nabla_x L(x_k,\, \mu_k) + \nabla_x^2 L(x_k,\, \mu_k)\delta_x + \nabla_{x\mu}^2 L(x_k,\, \mu_k)\delta_\mu = 0 \tag{4.16}$$

$$c_j(x_k + \delta_x) \approx c_j(x_k) + \delta_x^T \nabla_x c_j(x_k) \geq 0 \text{ pour } j = 1, 2, \ldots, q \tag{4.17}$$

$$\mu_{k+1} \geq 0 \tag{4.18}$$

et

$$[c_j(x_k) + \delta_x^T \nabla_x c_j(x_k)](\mu_{k+1})_j = 0 \quad \text{pour } j = 1, 2, \ldots, q \tag{4.19}$$

Le Lagrangien dans ce cas est définie par

$$L(x, \mu) = f(x) - \sum_{j=1}^{q} \mu_j c_j(x) \tag{4.20}$$

D'où

$$\nabla_x L(x_k,\, \mu_k) = \nabla_x f(x_k) - \sum_{j=1}^{q} (\mu_k)_j \nabla_x c_j(x_k) = g_k - A_k^T \mu_k$$

$$\nabla_x^2 L(x_k,\, \mu_k) = \nabla_x^2 f(x_k) - \sum_{j=1}^{q} (\mu_k)_j \nabla_x^2 c_j(x_k) = Y_k \tag{4.21}$$

et

$$\nabla_{x\mu}^2 L(x_k,\, \mu_k) = -A_k^T$$

où A_k est le Jacobien des contraintes au point x_k

$$\begin{bmatrix} \nabla_x^T c_1(x_k) \\ \nabla_x^T c_2(x_k) \\ . \\ . \\ . \\ \nabla_x^T c_q(x_k) \end{bmatrix} \tag{4.22}$$

Les conditions de KKT approchées dans les équations 4.16, 4.17, 4.18 et 4.19 peuvent être exprimées par

$$Y_k \delta_x + g_k - A_k^T \mu_{k+1} = 0 \tag{4.23}$$

$$A_k \delta_x \geq -c_k \tag{4.24}$$

$$\mu_{k+1} \geq 0 \tag{4.25}$$

$$(\mu_{k+1})_j (A_k \delta_x + c_k)_j = 0 \quad \text{pour } j = 1, 2, \ldots, q \tag{4.26}$$

où

$$c_k = [c_1(x_k)\ c_2(x_k)\ \ldots\ c_q(x_k)]^T \tag{4.27}$$

Etant donné un couple $(x_k,\ \mu_k)$, les équations 4.23, 4.24, 4.25 et 4.26 peuvent être interprétées comme les conditions de KKT exactes du problème QP suivant

$$\begin{array}{l} \text{minimiser } \frac{1}{2}\delta^T Y_k \delta + \delta^T g_k \\ \text{sous la contrainte : } \mathbf{A}_k \delta \geq -c_k \,. \end{array} \tag{4.28}$$

Si δ_x est une solution régulière du sous-problème QP 4.28 dans le sens où les gradients de ses contraintes sont actives au point x_k, alors l'équation 4.23 peut être ecrite sous la forme

$$Y_k \delta_x + g_k - A_{ak}^T \hat{\mu}_{k+1} = 0 \,,$$

où les lignes de A_{ak} sont ceux de A_k satisfaisant l'égalitée $(A_k \delta_x + c_k)_j = 0$ et $\hat{\mu}_{k+1}$ est le multiplicateur de Lagrange associé. Donc $\hat{\mu}_{k+1}$ peut être calculé par

$$\hat{\mu}_{k+1} = (A_{ak} A_{ak}^T)^{-1} A_{ak} (Y_k \delta_x + g_k) \tag{4.29}$$

La méthode SQP pour les problèmes de type inégalité peut être implémenté par l'algorithme suivant.

Algorithme

1. Poser $\{x, \mu\} = \{x_0, \mu_0\}$, $k = 0$ et initialiser une tolérence ϵ.
2. A l'itération k évaluer Y_k, A_k, g_k et c_k en utilisant respectivement les relations 4.21, 4.22, 4.8 et 4.27.
3. Résoudre le problème QP dans l'équation 4.28 pour δ_x et calculer le multiplicateur de Lagrange $\hat{\mu}_{k+1}$ en utilisant la relation 4.29.
4. Poser $x_{k+1} = x_k + \delta_x$. Si $\|\delta_x\| \leq \epsilon$ alors $x^* = x_{k+1}$; sinon, poser $k = k + 1$ et aller à l'étape 2.

4.4 SQP modifiée

Dans cette section, on présente deux modifications de l'algorithme SQP. Dans un premier temps, on va décrire une méthode de recherche linéaire proposée dans [23] qui permet aux deux algorithme précédents de converger avec un point initial arbitraire. La deuxième modification est basée sur l'approximation du Lagrangien en utilisant la formule BFGS définie dans [46].

4.4.1 Méthode de recherche linéaire

Considérons le problème avec contraintes 4.14, un pas de recherche linéaire peut être introduit dans l'algorithme de la section précédente, en générant l'itération $(k+1)$ par

$$x_{k+1} = x_k + \alpha_k \delta_x \tag{4.30}$$

où δ_x est la solution du sous-problème QP 4.28, et α_k est un scalaire obtenu par une recherche linéaire. Han [23] a proposé que le scalaire α_k est obtenu en minimisant la fonction de *pénalité*

$$\psi_h(\alpha) = f(x_k + \alpha\delta_x) + r \sum_{j=1}^{q} \max[0,\, -c_j(x_k + \alpha\delta_x)] \tag{4.31}$$

où r est un paramètre fixé. Une autre méthode pour détérminer α_k a été introduite par Powel [46], dans cette méthode une recherche linéaire est appliquée au Lagrangien dans l'équation 4.20 avec $x = x_k + \alpha\delta_x$ et $\mu = \mu_{k+1}$ pour obtenir la fonction à minimiser

$$\psi_p(\alpha) = f(x_k + \alpha\delta_x) - \sum_{j=1}^{q} (\mu_{k+1})_j c_j(x_k + \alpha\delta_x) \tag{4.32}$$

4.4.2 SQP avec approximation du Hessien

On approche le Lagrangien définie dans 4.20 par la formule de BFGS, ce qui donnera

$$Y_{k+1} = Y_k + \frac{\gamma_k \gamma_k^T}{\delta_x^T \gamma_k} - \frac{Y_k \delta_x \delta_x^T Y_k}{\delta_x^T Y_k \delta_x} \tag{4.33}$$

où $\delta_x = x_{k+1} - x_k$ et

$$\gamma_k = \nabla_x L(x_{k+1},\, \mu_{k+1}) - \nabla_x L(x_k,\, \mu_{k+1}) = (g_{k+1} - g_k) - (A_{k+1} - A_k)^T \mu_{k+1} \tag{4.34}$$

Si Y_k est définie positive, alors Y_{k+1} obtenu par l'équation 4.33 est définie positive si et seulement si

$$\delta_x^T \gamma_k > 0 \tag{4.35}$$

mais cette condition n'est pas toujours satisfaite, pour remédier à ce problème Powel [46] a suggéré de replacer γ_k dans l'équation 4.33 par

$$\eta_k = \theta\gamma_k + (1-\theta) Y_k \delta_x \tag{4.36}$$

où γ_k est donnée par 4.34 et θ est définie par

$$\theta = \begin{cases} 1 & \text{si } \delta_x^T \gamma_k \geq 0.2 \delta_x^T Y_k \delta_x \\ \dfrac{0.8 \delta_x^T Y_k \delta_x}{\delta_x^T Y_k \delta_x - \delta_x^T \gamma_k} & \text{sinon} \end{cases} \tag{4.37}$$

En utilisant la méthode de recherche linéaire et la formule de BFGS à l'algorithme décrit dans la section précédente, on obtient l'algorithme suivant qu'on peut appeler SQP modifiée pour les problèmes avec contraintes de type inégalité.

Algorithme

1. Poser $\{x, \mu\} = \{x_0, \mu_0\}$, $k = 0$ et initialiser une tolérence ϵ, $Y_0 = I_n$.
2. A l'itération k évaluer A_k, g_k et c_k en utilisant respectivement les relations 4.22, 4.8 et 4.27.
3. Résoudre le problème QP dans l'équation 4.28 pour δ_x et calculer le multiplicateur de Lagrange $\hat{\mu}_{k+1}$ en utilisant la relation 4.29.
4. Calculer α_k en minimisant ou bien la fonction $\psi_h(\alpha)$ définie dans 4.31, ou bien la fonction $\psi_p(\alpha)$ définie dans 4.32.
5. Poser $\delta_x = \alpha_k \delta_x$ et $x_{k+1} = x_k + \delta_x$.
6. Si $\|\delta_x\| \leq \epsilon$ alors $x^* = x_{k+1}$; sinon, aller à l'étape 7.
7. Evaluer γ_k, θ et η_k en utilisant respectivement les relations 4.34, 4.37 et 4.36. Calculer Y_{k+1} en utilisant 4.33.
   $k = k + 1$ et aller à l'étape 2.

4.5 SQP pour des problèmes avec des contraintes mixtes

Dans cette section, on va developper la méthode SQP pour des problèmes avec contraintes de type égalité et inégalité. Pour cela considérons le problème suivant

$$x^* = \text{argmin}\, \{f(x) : x \in C\} \tag{4.38}$$

$$C = \{x \in I\!R^n \,/\, a_i(x) = 0,\, i = 1, \ldots, p,\; c_j(x) \geq 0,\, j = 1, \ldots, q\}. \tag{4.39}$$

où $f : I\!R^n \longrightarrow I\!R$, $a_i : I\!R^n \longrightarrow I\!R,\ i = 1, \ldots, p$ et $c_j : I\!R^n \longrightarrow I\!R,\ j = 1, \ldots, q$ sont des fonctions régulières continues et leurs dérivées secondes sont aussi continues.
Soit $\{x_k, \lambda_k, \mu_k\}$ l'itération k et $\{\delta_x, \delta_\lambda, \delta_\mu\}$ un ensemble de vecteurs tel que l'itération $k+1$ est définie par $\{x_{k+1}, \lambda_{k+1}, \mu_{k+1} = \{x_k + \delta_x, \lambda_k + \delta_\lambda, \mu_k + \delta_\mu\}\}$. Les conditions de KKT suivantes doivent être satisfaites à l'itération $k+1$

$$\nabla_x L(x, \lambda, \mu) = 0$$

$$a_i(x) = 0 \quad \text{pour } i = 1, 2, \ldots, p$$

$$c_j(x) \geq 0 \quad \text{pour } j = 1, 2, \ldots, q$$

$$\mu \geq 0$$

$$\mu_j c_j(x) = 0 \quad \text{pour } j = 1, 2, \ldots, q\,,$$

où le Lagrangien $\nabla_x L(x, \lambda, \mu)$ est définie par

$$\nabla_x L(x, \lambda, \mu) = f(x) - \sum_{i=1}^{p} \lambda_i a_i(x) - \sum_{j=1}^{q} \mu_j c_j(x)\,.$$

En utilisant la même procédure que dans la section précédente, on obtient les conditions de KKT approchées

$$Z_k \delta_x + g_k - A_{ek}^T \lambda_{k+1} - A_{ik}^T \mu_{k+1} = 0 \tag{4.40}$$

$$A_{ek} \delta_x = -a_k \tag{4.41}$$

$$A_{ik} \delta_x \geq -c_k \tag{4.42}$$

$$\mu_{k+1} \geq 0 \tag{4.43}$$

$$(\mu_{k+1})_j (A_{ik} \delta_x + c_k)_j = 0 \quad \text{pour } j = 1, 2, \ldots, q \tag{4.44}$$

où

$$Z_k = \nabla_x^2 f(x_k) - \sum_{i=1}^{p} (\lambda_k)_i \nabla_x^2 a_i(x_k) - \sum_{j=1}^{q} (\mu_k)_j \nabla_x^2 c_j(x_k) \tag{4.45}$$

$$g_k = \nabla_x f(x_k) \tag{4.46}$$

$$A_{ek} = \begin{bmatrix} \nabla_x^T a_1(x_k) \\ \nabla_x^T a_2(x_k) \\ . \\ . \\ . \\ \nabla_x^T a_p(x_k) \end{bmatrix} \tag{4.47}$$

$$A_{ik} = \begin{bmatrix} \nabla_x^T c_1(x_k) \\ \nabla_x^T c_2(x_k) \\ . \\ . \\ . \\ \nabla_x^T c_q(x_k) \end{bmatrix} \tag{4.48}$$

a_k et c_k sont données respectivement par les équations 4.9 et 4.27.
Etand donné (x_k, λ_k, μ_k), les équations 4.40-4.44 peuvent être interprétées comme les conditions de KKT exactes du problème QP

$$\begin{aligned} &\text{minimiser } \tfrac{1}{2}\delta^T Z_k \delta + \delta^T g_k \\ &\text{sous les contrainte : } \mathbf{A}_{ek}\delta = -a_k \\ &\qquad\qquad\qquad\quad \mathbf{A}_{ik}\delta \geq -c_k \end{aligned} \tag{4.49}$$

Soit A_{aik} la matrice composée des lignes de A_{ik} satisfaisants l'égalité $(A_{ik}\delta_x + c_k)_j = 0$ et $\hat{\mu}_{k+1}$ le multiplicateur de Lagrange associé. λ_{k+1} et $\hat{\mu}_{k+1}$ peuvent être calculés par

$$\begin{bmatrix} \lambda_{k+1} \\ \hat{\mu}_{k+1} \end{bmatrix} = (A_{ak}A_{ak}^T)^{-1} A_{ak}(Z_k\delta_x + g_k) \tag{4.50}$$

où

$$A_{ak} = \begin{bmatrix} A_{ek} \\ A_{aik} \end{bmatrix}$$

Si δ_x est la solution du problème QP 4.49, alors

$$x_{k+1} = x_k + \alpha_k \delta_x \tag{4.51}$$

où α_k est déterminée en minimisant la fonction de merite

$$\psi(\alpha) = f(x_k + \alpha\delta_x) + \beta \sum_{i=1}^{p} a_i^2(x_k + \alpha\delta_x) - \sum_{j=1}^{q} (\mu_{k+1})_j c_j(x_k + \alpha\delta_x) \tag{4.52}$$

où β est un positif scalaire suffisamment grand. Une fois x_{k+1} est determiné, un Hessien approché Z_{k+1} est calculé en utilisant la formule BFGS

$$Z_{k+1} = Z_k + \frac{\eta_k \eta_k^T}{\delta_x^T \eta_k} - \frac{Z_k \delta_x \delta_x^T Z_k}{\delta_x^T Z_k \delta_x} \tag{4.53}$$

où

$$\eta_k = \theta \gamma_k + (1-\theta) Z_k \delta_x \tag{4.54}$$

$$\gamma_k = (g_{k+1} - g_k) - (A_{e,k+1} - A_{e,k})^T \lambda_{k+1} - (A_{i,k+1} - A_{i,k})^T \mu_{k+1} \tag{4.55}$$

$$\theta = \begin{cases} 1 & \text{si } \delta_x^T \gamma_k \geq 0.2 \delta_x^T Z_k \delta_x \\ \dfrac{0.8 \delta_x^T Z_k \delta_x}{\delta_x^T Z_k \delta_x - \delta_x^T \gamma_k} & \text{sinon} \end{cases} \tag{4.56}$$

Cette méthode peut, donc, être implémenté selon l'algorithme suivant

Algorithme

1. Poser $\{x, \lambda, \mu\} = \{x_0, \lambda_0, \mu_0\}$, $k = 0$ et initialiser une tolérence ϵ, $Z_0 = I_n$.
2. A l'itération k évaluer g_k, A_{ek}, A_{ik}, a_k et c_k en utilisant respectivement les relations 4.46, 4.47, 4.48, 4.9 et 4.27.
3. Résoudre le problème QP dans l'équation 4.49 pour δ_x et calculer le multiplicateur de Lagrange $\hat{\mu}_{k+1}$ en utilisant la relation 4.50.
4. Calculer α_k en minimisant la fonction $\psi(\alpha)$ définie dans 4.52.
5. Poser $\delta_x = \alpha_k \delta_x$ et $x_{k+1} = x_k + \delta_x$.
6. Si $\|\delta_x\| \leq \epsilon$ alors $x^* = x_{k+1}$; sinon, aller à l'étape 7.
7. Evaluer γ_k, θ et η_k en utilisant respectivement les relations 4.55, 4.56 et 4.54. Calculer Z_{k+1} en utilisant 4.53.
   $k = k + 1$ et aller à l'étape 2.

4.6 Note sur SQP perturbée

Même si la méthode SQP s'est avérée efficace pour les problèmes non linéaires, et mieux que les méthodes basées sur le gradient, mais elle ne trouve pas l'optimum global dans le cas des problèmes non convexes, ce qui nous

a poussé à la perturber aléatoirement comme on l'a fait dans le cas du gradient projeté discuté dans le chapitre précédent. Pour perturber la méthode SQP, l'idée est de choisir une suite aléatoire R_k construite comme dans le chapitre précédent, puis dans l'étape 5 de l'algorithme précédent faire $xt_{k+1} = x_k + \delta_x + R_k$, et pour rendre le point réalisable, on le corrige vers la frontière en utilisant la méthode 3.30, ce qui nous donnera un point x_k réalisable, et on termine l'algorithme normalement. le théorème 3.3 peut être appliqué à la méthode SQP perturbée pour montrer sa convergence.

4.7 Résultats numériques

Dans cette section, on donnera des résultats numériques de 6 exemples académiques implémentés en utilisant la méthode SQP et la méthode SQP perturbée, pour comparer les deux algorithmes.

4.7.1 Problèmes

Les problèmes choisis pour tester la méthode SQP sont les problèmes 1-6 cités dans la section 3.5 du chapitre Gradient projeté perturbé.

4.7.2 Résultats numériques

Dans les tableaux suivants les résultats numériques des problèmes définis dans la section (4.7.1), en utilisant la méthode SQP et la méthode SQP perturbé. PN est le numéro du problème, x^* le vecteur optimal et $f(x^*)$ la valeur optimal de la fonction objectif.
Tous les programmes ont été implémenté en Matlab 6.5.1, sous Windows XP et Pentium (R) 4 CPU 1400 MHZ.

PN	x^*	f(x^*)	temps(sec)
1	$(0,\ 3.5691,\ 1.3315,\ 5.3272,\ 4.8591,\ 0.6177,\ 1.5281,\ 0.9996)$	39.19	1.0470
2	$(78,\ 33,\ 30,\ 31,\ 42)$	10484	0.6400
3	$(40,\ 40,\ 60,\ 0.1)$	766170	2.33
4	$(-2.2863,\ -1.3694)$	4.3130	0.32
5	$(1,\ 0.7090,\ -0.7052)$	-7.3064	0.35
6	$(-2,\ -1)$	1.7333	0.01

TABLE 4.1 – Résultats numériques de la méthode SQP

PN	x^*	f(x^*)	temps(sec)
1	$(0,\ 2.3747,\ 1.0050,\ 5.3820,\ 6.7589,\ 0.7845,\ 1.5504,\ 0.1545)$	35.099	2.4126
2	$(78,\ 33,\ 29.997,\ 45,\ 36.772)$	10123	1.647
3	$(40,\ 40,\ 70,\ 0.5)$	538510	6.22
4	$(-0.0555,\ -0.8379)$	0.6617	0.67
5	$(2-\sqrt{2},\ -\sqrt{2},\ -\sqrt{2})$	-12.1716	0.86
6	$(-1.7029,\ -0.7914)$	-0.2152	0.18

TABLE 4.2 – Résultats numériques de la méthode SQP perturbée

Interprétation des résultats

Il s'est avéré que la méthode SQP perturbé donne de meilleurs résultats que la méthode SQP au niveau de la précision des résultats obtenus.
Néanmoins, dans certains cas, l'optimum global n'est pas atteint.
L'autre remarque est que les méthodes SQP et SQP perturbé donnent de meilleurs résultats que le gradient projeté et le gradient projeté perturbé au niveau de la précision de la solution et du temps de calcul.

4.8 Conclusion

La méthode SQP est une méthode locale, basée sur une recherche linéaire et une approche Lagrangienne du problème.
Les essais numériques montrent que le rôle des perturbations est essentiel : lorsque celles-ci ne sont pas présentes, le résultat se dégrade. Cette observation tend à confirmer le choix d'une approche hybride.

Deuxième partie

Algorithmes évolutionnaires

Chapitre 5

Algorithme génétique

5.1 Introduction

La plupart des méthodes d'optimisation sont des méthodes locales qui ne peuvent pas converger vers un optimum global dans le cas d'optimisation non convexe, aussi ils ont l'inconvénient de boucler, et revenir sans cesse au même point. Ces algorithmes (de la famille du gradient ou Newton-Raphson) sont basés sur l'évaluation du gradient pour déterminer la direction de recherche de l'optimum.
Goffe [20] compare cette situation à un aveugle qui veut trouver le sommet d'une montagne :
ce dernier reconnait le sol seulement avec ses pieds ; si le sol est régulier et le point de départ est bon, alors il trouvera le sommet. Mais ces deux conditions sont rarement accomplies.

Pour éviter ces inconvénients des auteurs ont pensé à utiliser des méthodes stochastiques comme l'algorithme génétique et le récuit simulé ([21], [28], [51]) qui ne requièrent aucune hypothèse sur la fonction objectif ou sur les contraintes.
L'algorithme génétique est inspiré de la théorie génétique élaborée par Charles Darwin [11]. Le vocabulaire utilisé est donc similaire à celui de la théorie génétique et évolutionnaire. On utilisera donc les termes : *individus* (poids des solutions), *population*, *gènes* (variables), *chromosome*, *parents*, *croisement*, *mutation*,....
En résumé, la définition de l'algorithme génétique se présente comme suit :

Un algorithme génétique est un algorithme stochastique itératif qui opère sur un ensemble de points, partant d'une population initiale. Il est construit en utilisant 3 opérateurs :

- croisement,
- mutation,
- sélection.

Nous aborderons l'algorithme génétique utilisé pour résoudre un problème d'optimisation, puis nous le testerons sur différents problèmes numériques.

5.2 Généralités

Historique

La programmation évolutionnaire a été introduite en 1960 par I. Rechenberg [48]. Son idée a été développée par d'autres chercheurs.
Les algorithmes génétiques ont été inventés et développés par John Holland dans son livre [28].
Goldberg [21] est le premier à utiliser l'algorithme génétique pour résoudre des problèmes pratiques.

Principes

L'algorithme génétique consiste à évoluer, à partir d'une population initiale, un ensemble de points dans l'espace de recherche pour trouver l'optimum.
Ce procédé est appliqué à une population d'une taille constante N, de telle manière que toutes les générations successives soient aussi de taille N.
Pour faire évoluer cette population de la génération k à la génération $k+1$, des opérations sont appliquées à tous les individus de la génération k :

- Une sélection d'individus de la génération k est effectuée selon le critère d'optimisation : on veut priviligier la reproduction des "bons" éléments au lieu des "mauvais".
- Des opérateurs d'exploration de l'espace sont alors utilisés :

  - L'opérateur de croisement est appliqué avec une probabilité P_c à deux éléments de la génération k qui seront transformés en conséquence en deux autres éléments qui vont les remplacer dans la génération $k+1$.
  - L'opérateur de mutation est alors appliqué avec une probabilité P_m pour modifier les individus.
- Arrêt de l'algorithme génétique.
  - Arrêter après un nombre fixé d'itérations (ou de générations).
  - Arrêter quand la population n'évolue plus suffiseamment.

5.3 Opérateurs de l'algorithme génétique

L'algorithme génétique simple produit une *génération* lorsque les opérateurs définis ci-dessus ont été appliqués simultanément aux individus de la population, selon la séquence suivante :

- décodage des solutions,
- évaluation des solutions,
- sélection des individus,
- croisement et mutation.

De génération en génération, la taille de la population reste constante. Lors d'une génération, la totalité d'une population peut être remplacée par ses descendants. L'algorithme génétique est alors dit "générationnel" (generational). Dans le cas contraire, on se trouve, alors, en présence d'un algorithme "stationnaire" (steady state).

Les opérateurs de reproduction, introduisent continuellement de la diversité dans la population.
Pour cette raison, une bonne solution, codée par un génotype présent à une génération donnée, peut disparaître et éventuellement réapparaître dans les générations suivantes. Un algorithme génétique simple ne converge donc pas au sens habituel du terme, où il existerait une certitude de disposer des bonnes solutions dans la population à partir d'un nombre de générations suffisamment important.
En fait, les paramètres de l'algorithme étant fixés, la population atteint un état stationnaire où chaque présence du génotype possible a une probabilité non

nulle, constante, mais inconnue a priori.
Ce qu'il faut espérer dans cette situation, c'est que de bonnes solutions ont une forte probabilité d'être découvertes. Ce n'est pas toujours le cas, et c'est cela qui distingue les problèmes algorithmes génétiques-faciles des problèmes difficiles.
L'algorithme génétique converge quand la plupart des individus d'une population sont identiques. Toutefois, cette définition ne convient plus lorsque l'algorithme est configuré pour exhiber les différentes solutions d'un problème à une génération donnée.
Pratiquement, l'algorithme effectue autant de générations que nécessaire jusqu'à ce qu'un critère d'arrêt, lié par exemple à la qualité des solutions obtenues, soit respecté.

5.3.1 Codage

Il y'a deux types de difficultés dans le choix du codage.
D'une part, il doit être capable de s'adapter au problème ;
et d'autre part il doit garder les propriétées de nouveaux éléments générés par les opérateurs de recherche.
Le codage binaire choisi en vue d'implémenter l'algorithme génétique a démontré son efficacité.

Conversion décimale binaire

On décompose le nombre à convertir en un ensemble de bits, par des divisions succéssives par deux.
Le reste de la première division est le bit de droite, le reste de la deuxième division est le bit suivant, et ainsi de suite jusqu'à avoir un quotient nul. Le reste de cette dernière division est le bit de gauche.
Exemple
10=00001010
15=00001111
254=11111110

Code du complément à deux :

Ce code a besoin de deux opérations pour représenter un nombre :
- On calcul le complément à 1 (C to 1).
- On ajoute 1 au complément à 1.

Complément à 1

Le complément à 1 est un nombre binaire obtenu en inversant les états logiques des bits d'un autre nombre binaire.

$$11110000 \longrightarrow 00001111$$

Dans ce travail, on code chaque chromosome par une chaine binaire, selon le format :

Exposant (E)	Signe	Mantisse (M)

- L'exposant est codé en complément à 2.
- Signe=1 nombre négatif.
  Signe=0 nombre positif.
- La mantisse est codée en binaire.

Exemple

N=4,75=00000100,11000000 en binaire
=0.10011000000 $\times 2^3$

M=0.10011000000
E=3=00000011 en complément à deux.

5.3.2 Fonction d'adaptation

Les individus les mieux adaptés, c'est-à-dire capables de mieux effectuer les tâches nécessaires à leur survie, se reproduisent à des taux plus élevés.
Alors que les individus les moins adaptés se reproduisent à des taux plus faibles.

Ce sont les principes de survie et reproduction décrits par Charles Darwin dans [11].
Il s'avère, alors, qu'une population ayant une grande variété engendrera, de génération en génération, une contenance des individus dont le génotype se traduit par une meilleure adaptation.
Pour celà il nous faut une fonction dite d'adaptation qui va être dans notre cas la fonction objectif elle même.

5.3.3 Croisement

Après avoir décidé du codage à utiliser, on peut passer à l'opération de croisement.
Il est appliqué à deux éléments de la population choisis aléatoirement avec une probabilité P_c.
Dans le cas d'un codage réel, on peut choisir comme opérateur de croisement une transformation affine, par exemple si x et y sont les parents on peut générer des enfants x_1 et y_1 comme suit :

$$x_1 = \alpha x + \beta y + \gamma \ , \quad y_1 = \alpha_1 x + \beta_1 y + \gamma_1$$

où $\alpha,\ \beta,\ \gamma,\ \alpha_1,\ \beta_1,\ \gamma_1$ sont des constantes choisies aléatoirement.
Dans le cas d'un codage binaire, base de notre implémentation de l'algorithme génétique, il nous est possible de faire le croisement en choisissant aléatoirement un point de coupure et de faire l'opération suivante : (| est le point de croisement)

parent 1	1100|01100100010
parent 2	0101|11000111011
enfant 1	1100|11000111011
enfant 2	0101|01100100010

5.3.4 Mutation

L'opérateur de mutation permet d'éviter une convergence prématurée vers un optimum local en maintenant une diversité de solution.

Pour l'appliquer dans le cas du codage binaire usité, par un choix au hasard d'un bit du chromosome et de modification de sa valeur.
Dans le cas d'un codage réel, on peut utiliser une méthode d'optimisation classique (voir chapitre Hybridation).
La mutation ne doit pas être appliquée systématiquement, mais en fonction d'une probabilité P_m, paramètre de la simulation.

5.3.5 Sélection

Les chromosomes sont sélectionnés à partir de la population pour être des parents pour des enfants produits par croisement ou mutation.
Pour choisir ses éléments il y'a plusieurs méthodes : le choix par tournois, l'élitisme, la sélection de Boltzman ou la roulette .... Lors de ce labeur, nous avons choisi la méthode d'élitisme.

Elitisme

Une stratégie élitiste consiste à conserver dans la population, d'une génération à l'autre, au moins l'individu ayant la meilleure adaptation.
Il apparaît qu'une telle stratégie améliore considérablement les performances de l'algorithme génétique pour certaines classes de fonctions.
En fait, une stratégie élitiste favorise l'exploitation des meilleures solutions, à tràvers une recherche locale accentuée.

5.4 Algorithme

L'algorithme génétique qu'on a implémenté se présente comme suit :
Une population initiale est générée aléatoirement de N individus P_0.
Le processus suivant y est appliqués à l'itération k :

1. Codage de chaque individu de la population P_k.
2. Croisement de tous les individus de la population deux par deux avec une probabilité P_m, on aura N enfants notés C_k.
3. Mutation de tous les individus de la population, on aura N éléments notés M_k.

4. Choisir les N meilleurs éléments de l'ensemble $P_k \bigcup C_k \bigcup M_k$ c.à.d. ceux qui ont une valeur inférieure de la fonction objectif, et de les mettre dans la population P_{k+1}.
5. Si le test d'arrêt est vérifié arrêter, sinon revenir à l'étape 1.

Nous avons choisi, comme test d'arrêt dans notre implémentation un nombre fini d'itérations.

5.5 Problèmes avec contraintes

Les problèmes d'optimisation issus du monde réel doivent souvent respecter un certain nombre de contraintes. Celles-ci se traduisent par un ensemble de relations que doivent satisfaire les variables de la fonction dont on cherche l'optimum. Ces relations sont généralement exprimées comme un ensemble de q inégalités et m égalités.
Or, les opérateurs de recherche standards décrits ci-dessus engendrent des génotypes de façon aveugle, ne tenant pas compte des contraintes, et qui peuvent correspondre à des phénotypes irréalisables.
Pour imposer le respect des contraintes d'un problème, plusieurs approches agissant sur les différents opérateurs de l'algorithme génétique sont utilisables.

- Une première solution consiste à ne calculer la fonction d'adaptation que dans l'espace réalisable des phénotypes. Les individus de l'espace irréalisable se voient affecter une adaptation nulle qui empêchera leur reproduction. Il s'agit de la méthode de la peine de mort (death penalty method). Cette méthode qui a l'avantage d'être simple s'est avérée très peu performante. Elle peut être assimilée à une méthode par pénalisation avec pénalité infinie.
- Une solution plus élaborée et plus efficace consiste à calculer une valeur d'adaptation aussi bien dans l'espace réalisable que dans l'espace irréalisable et, dans ce dernier cas, de lui ajouter une pénalité (pour une recherche de minimum) dont la valeur donne une indication sur le nombre et l'importance des contraintes violées. Toutefois, de faibles pénalités risquent de faire apparaître des individus irréalisables dont l'adaptation est meilleure que celle d'individus réalisables. A l'inverse, de fortes pénalités associées

à une forte production d'individus iréalisables risquent de faire converger prématurément l'algorithme génétique vers une solution réalisable mais de pauvre qualité.
- Une autre approche consiste à utiliser des algorithmes de réparation lors du décodage, pour transformer un phénotype qui devrait être irréalisable en phénotype réalisable. Cependant, cette solution consomme généralement beaucoup de puissance de calcul et sa mise en oeuvre est souvent délicate.
- Une quatrième possibilité revient à ne produire que des génotypes codant des phénotypes réalisables. Cela conduit à remplacer les opérateurs de recherche standards par des opérateurs dépendants du problème. Une telle solution permet de décrire le problème posé de façon plus aisée, elle élimine quasiment la phase de décodage, mais elle implique que les opérateurs génétiques de recherche soient spécialisés.
- D'autres méthodes existent, comme ramener un problème d'optimisation sous contraintes en un problème d'optimisation multi-critères sans contrainte, où les contraintes sont considérées des fonctions objectif plus prioritaires que le critère à optimiser.

Dans notre cas, on a utilisé la méthode de correction 3.30 utilisée dans le chapitre "Gradient projeté perturbé", qui consiste à ramener un point irréalisable vers la frontière.

5.6 Résultats numériques

5.6.1 Problèmes sans contraintes

Les problèmes sans contraintes utilisés pour l'évaluation de l'algorithme génétique sont ceux définit dans la table 2.4.1 du chapitre 2.

5.6.2 Problèmes avec contraintes

Les problèmes avec contraintes utilisés pour l'évaluation de l'algorithme génétique sont :
problèmes 1-4 : problèmes 1-4 du chapitre 2.

problèmes 5-10 : problèmes 1-6 du chapitre 3.

5.6.3 Résultats numériques

Dans les tableaux suivants les résultats numériques des problèmes définis ci-dessus. Tous les programmes ont été implémenté en Matlab 6.5.1, sous Windows XP et Pentium (R) 4 CPU 1400 MHZ.

Numéro de la fonction	Valeur optimale	vecteur optimal	temps (sec)
1	2.5083	(0.9666,0.6388)	16.20
2	0.9999	(1.5612,1.0053)	16.34
3	0.9999	(4.7039,4.7135)	16.10
4	50.89	(0.7818,0.3234,0.2877)	24.10
5	0.9999	(0.1506,0)	15.96
6	-1.2284	(2.5922,1.5876)	16.43
7	-0.1691	(1.7868,1.3744)	16.28
8	-1.7441	(1.7433,2.0215)	16.23

TABLE 5.1 – Résultats numériques de l'algorithme génétique sans contraintes

Numéro de la fonction	Valeur optimale	Vecteur optimal	temps (sec)
1	2819	(1.2959,2.8291,2.6281,1.0844,1.3570)	42.87
2	-4.6839	(1.4022,1.7848,1.8102)	26.31
3	n.cvt		22.2
4	n.cvt		23.35
5	n.cvt		23.35
6	n.cvt		23.35
7	n.cvt		23.35
8	n.cvt		23.35
9	n.cvt		23.35
10	n.cvt		23.35

TABLE 5.2 – Résultats numériques de l'algorithme génétique avec contraintes

N.B : n.cvt veut dire non convergent.

Interprétation des résultats

Puisque le test d'arrêt que nous avons choisi pour l'implémentation de l'algorithme génétique est le nombre d'itérations constant, donc le nombre d'évaluation de la fonction objectif et des contraintes est aussi constant ; le temps de calcul est aussi constant car on a un même nombre d'itérations. L'algorithme génétique n'a pas convergé dans le cas avec contraintes vers une solution réalisable et dans le cas sans contraintes il a donné de mauvaises solutions du problème.
Tout cela nous emmène à conclure que l'algorithme génétique ne peut pas être exploité pour résoudre des problèmes d'optimisation complexes.

5.7 Conclusion

L'atout majeur de l'algorithme génétique est qu'il ne dépend aucunement d'une hypothèse sur la fonction objectif ou sur les contraintes, et en théorie s'applique à des situations très générales.
Cependant, en pratique, cet algorithme peut exiger un grand nombre d'évaluations de la fonction sous peine de conduire à des résultats de mauvaise qualité. Le couplage avec des algorithmes déterministes améliore nettement les résultats, ce qui tent à comporter le choix d'une approche hybride.

Chapitre 6

Recuit simulé

6.1 Introduction

Le recuit simulé est un algorithme qui a été conçus afin de trouver une solution optimale d'un problème d'optimisation conbinatoire.
Il a été mis au point par trois chercheurs de la société IBM, S. Kirkpatrick, C.D. Gelatt et M.P. Vecchi en 1983 [32].

L'idée du recuit simulé en optimisation combinatoire est de représenter numériquement une opération de recuit thermique.
Le principe en est le suivant :

On considère un système composé de N éléments.
A chaque configuration du système est associée une fonction à optimiser.
Si l'on recherche la configuration qui minimise la fonction, celle-ci joue le rôle de l'énergie.
Si l'on recherche au contraire un maximum, alors on prend comme énergie l'opposée de la fonction.
Les configurations du système peuvent être modifiées par des changements d'états discrets des composants du dit système.
Les systèmes qui nous intéressent sont ceux qui ont un grand nombre d'optimums.

Notre point de départ est une configuration aléatoire voire choisie astucieusement en fonction du problème.
A l'itération initiale, on fixe un paramètre de température en relation avec la

gamme des énergies accessibles au système.
On itère alors le processus suivant ; dit de Metropolis ou de Monte-Carlo :
on tire au sort une modification de la configuration actuelle qui change l'énergie du système.
Si l'énergie diminue, on effectue le changement.
Si l'énergie augmente, on effectue le changement avec une certaine probabilité.
Notons que La procédure de Monte-Carlo et celle des automates probabilistes, sont équivalentes du point de vue des états stationnaires obtenus.
Les automates probabilistes nécessitent un peu plus de calcul d'exponentielles et de tirage au sort.
On préfère donc Monte-Carlo dans les cas pratiques.

L'itération se poursuit tant que l'énergie du système diminue. Lorsque l'énergie reste stationnaire, on diminue un peu la température et l'on reprend le processus Monte-Carlo de décroissance de l'énergie.
On arrête lorsque les diminutions de température restent inefficaces.

6.2 Algorithme

L'algorithme du recuit simulé peut être implémenté comme suit :

1. (Déplacement) : partir de la solution courrante vers une nouvelle solution.
2. (Evaluation) : évaluer le coût de la nouvelle solution.
3. (Acceptation) : décider d'accepter ou de rejetter une nouvelle solution.
4. (Test d'arrêt) : si le test d'arrêt est vérifié arrêter sinon revenir à 1.

La différence entre le recuit simulé et les autres algorithmes est qu'il peut accepter une nouvelle solution, même si elle provoque une augmentation de l'énergie du système, avec une probabilité qui est généralement égale à

$$P = \exp\left(-\frac{\Delta E}{\theta}\right)$$

où θ est un paramètre lié à la température.
La fonction d'acceptation peut être implémentée comme suit :

Procédure ACCEPTATION (J,$\theta(k)$)
Début
  Si $\Delta J \leq 0$ **Alors**
    ACCEPTATION := vrai ;
  Sinon
    $C_k(\theta) := \exp\left(-\frac{\Delta E}{\theta(k)}\right)$
    Si aléatoire(0,1) $< C_k(\theta)$ **Alors**
      ACCEPTATION := vrai ;
    Sinon
      ACCEPTATION := faux ;
    Fin Si
  Fin Si
Fin Procédure ACCEPTATION

6.3 Convergence de l'algorithme

Dans cette section, on va citer un théorème de convergence. Pour plus de détails et preuves des résultats, nous vous invitons à voir [51].
Soit $J(p_1, p_2, \ldots, p_n)$ la fonction coût. Soit $G(V, E)$ son graphe, on peut construire un graphe direct $G(V, E)$ correspondant au problème d'optimisation donné. Soit T la température minimale que le recuit simulé a accédé. Et soit

$$\Delta = \max_{i \in V, j \in N(i)} \{J(i) - J(j)\}$$

. On note respectivement le degrée et le diamètre de $G(V, E)$ par d D.

Lemme 6.1 *Si X est un état de V , donc l'espérance du nombre d'itération avant que l'optimum global soit visité est* $\leq \left(\frac{1}{d}\exp\left(-\Delta/T\right)\right)^{-D}$.

Théorème 6.1 *Le recuit simulé converge dans un temps* $\leq 2k\left[d\exp\left(-\Delta/T\right)\right]^{D}$, *avec une probabilité* $\geq (1 - 2^{-k})$.

6.4 Problème du voyageur de commerce

Pour des problèmes NP complets d'optimisation (comme le problème du voyageur de commerce), on ne connait pas d'algorithme polynomial permettant une résolution de façon optimale.
On va donc chercher une solution approchée de cette optimum en utilisant des heuristiques.
Le recuit simulé est un algorithme basé sur une heuristique permetant la recherche de solutions à un problème donnée et qui a montré son efficacité pour résoudre le problème du voyageur de commerce.
Dans cette section on va définir ce problème et donner un résultat de convergence du recuit.

6.4.1 Définition

Un voyageur de commerce doit se rendre dans plusieurs villes.
Il connaît les distances entre toutes les villes et voudrait déterminer l'ordre dans lequel il doit faire ses visites pour minimiser la distance parcourue (et donc le coût de la tournée).
On numérote les villes de 1 à d.
On notera $\delta(a, b)$ la distance entre les villes a et b.
Ces distances sont connues. A une tournée dans les d villes, on associe (bijectivement) une permutation circulaire σ de $\{1, \ldots, d\}$.
On note f l'application qui à une permutation σ associe

$$f(\sigma) = \sum_{i=0}^{d-1} \delta(\sigma^i(1), \sigma^{i+1}(1)).$$

La quantité $f(\sigma)$ représente la longueur totale de la tournée dans l'ordre spécifié par σ, avec retour au point de départ.
Le problème est donc de trouver une permutation σ qui minimise $f(\sigma)$.
Il s'agit d'un problème d'optimisation sur l'ensemble E des permutations de d éléments. Il est fini mais devient vite très grand avec d.
Evaluer f sur toutes les permutations prendrait un temps immense.

6.4.2 Structures de graphe sur l'ensemble des permutations

On notera E l'ensemble des permutations de $\{1, \ldots, d\}$. Soit A une partie de E^2 telle que si $(\sigma, \sigma^{'}) \in A$ alors $(\sigma^{'}, \sigma) \in A$. Si $(\sigma, \sigma^{'}) \in A$, on dit que σ et $\sigma^{'}$ sont voisins et l'on note $\sigma \approx \sigma^{'}$. On a ainsi construit un graphe non orienté (E, A) où E est l'ensemble des sommets et A l'ensemble des arêtes.
Si σ et $\sigma^{'}$ sont deux éléments de E, un chemin de longueur n de σ à $\sigma^{'}$ est une suite $\sigma_0 = \sigma, \sigma_1, \ldots, \sigma_{n-1}, \sigma_n = \sigma^{'}$ de points de E tels que deux points consécutifs sont voisins.
Dire que le graphe est connexe signifie que pour tout couple de points de E, il existe un chemin de l'un vers l'autre.
La distance de σ à $\sigma^{'}$ est la plus petite longueur d'un chemin entre σ et $\sigma^{'}$. Le diamètre du graphe est la plus grande distance entre deux points de E. On le notera D.

6.4.3 Mesures de Gibbs

Définition et propriétés essentielles

On appelle mesure de Gibbs associée à la fonction d'énergie f à température $T > 0$ la loi de probabilité $v^T = (v_\sigma^T)_{\sigma \in E}$ telle que

$$\forall \sigma \in E \quad v_\sigma^T = \frac{1}{Z(T)} exp(-\frac{1}{T} f(\sigma))$$

où la constante de normalisation est

$$Z(T) = \sum_{i \in E} exp(-\frac{1}{T} f(i))$$

La proposition suivante explicite le comportement de la suite de mesures v^T en basse et haute températures : dans le premier cas elle converge vers la mesure uniforme sur les états de plus basse énergie, dans le second elle converge vers la mesure uniforme sur E.

Proposition 6.1 *Si $E_{min} = \{i \in E \ : \ \forall j \in E, \ f(i) \leq f(j)\}$, alors pour tout $\sigma \in E$:*

$$\lim_{T \to 0^+} v_\sigma^T = \frac{1}{|E_{min}|} \quad \textit{et} \quad \lim_{T \to \infty} v_\sigma^T = \frac{1}{|E|}.$$

6.4.4 Concept physique

On définie l'entropie de la loi de probabilité $\mu = (\mu_i)_{i \in E}$ par

$$H(\mu) = -\sum_{i \in E} \mu_i \log \mu_i$$

avec la convention $0 \log 0 = 0$.

Lemme 6.2 *L'entropie est toujours positive, maximale pour la loi uniforme sur E et minimale pour les mesures de Dirac.*

L'entropie est une mesure de l'incertitude, de l'aléa contenu dans la loi μ.
En physique, à chaque état i d'un système est associé une énergie, représentée dans notre problème par $f(i)$.
Un grand principe de la physique prédit que les distributions stables sont celles dont l'entropie est maximale.
La distribution μ sur E possède une énergie moyenne égale á

$$E_\mu(f) = \sum_{i \in E} f(i)\mu_i$$

Si l'on impose que cette énergie soit constante égale à Γ, peut-on trouver les distributions qui maximisent l'entropie ?
Pour qu'il en existe, il faut bien sûr que $\Gamma \in [\min f, \max f]$.
Supposons de plus que f ne soit pas constante (sinon la question n'est pas très intéressante).
Dans ce cas, il existe une seule mesure de probabilité répondant à la question, elle est appelée loi de Boltzmann.

Lemme 6.3 *Si f n'est pas constante, alors, à énergie moyenne constante égale à* $\Gamma \in]\min f, \max f[$*, la mesure de probabilité sur E qui maximise l'entropie est de la forme*

$$\forall i \in E \quad \mu_i = \frac{1}{Z_a} exp(-af(i)),$$

où $Z_a = \sum_{j \in E} exp(-af(j))$ *et a est l'unique solution de* $\Gamma = \sum_{i \in E} f(i)exp(-af(i))/Z_a$

6.5 Algorithme du recuit simulé et résultat de convergence

La méthode du recuit simulé tire son nom et son inspiration de la physique des matériaux et plus spécialement de la métallurgie.
Le recuit est une opération consistant à laisser refroidir lentement un métal pour améliorer ses qualités.
L'idée physique est qu'un refroidissement trop brutal peut bloquer le métal dans un état peu favorable, alors qu'un refroidissement lent permettra aux molécules de s'agencer au mieux dans une configuration stable.
C'est cette même idée qui est à la base du recuit simulé.
Pour éviter que l'algorithme ne reste piégé dans des minima locaux, on fait en sorte que la température $T = T(n)$ décroisse lentement en fonction du temps.
On peut montrer que la décroissance de T doit être de type logarithmique.
Le théorème ci-dessous [51] montre que l'on peut également choisir T constante par morceaux sur des paliers de longueur exponentielle.

Théorème 6.2 *Soit $h > 0$. On choisit T constante par morceaux :*

$$\forall k \in I\!N^*, \quad \forall n \in]\exp{(k-1)h}, \exp{kh}[, \quad T(n) = \frac{1}{k}.$$

Il existe $h^* > 0$ tel que pour tout $h > h^*$, l'algorithme du recuit converge, c'est à dire que

$$\lim_{n\to\infty} P(X_n \in E_{min}) = 1.$$

6.6 Résultats numériques

Dans cette section, on donnera des résultats numériques de 6 exemples académiques implémentés en utilisant la méthode SQP et la méthode SQP perturbée, pour comparer les deux algorithmes.

6.6.1 Problèmes

Les problèmes choisis pour tester l'algorithme du recuit simulé sont les problèmes 1-6 cités dans la section 3.5 du chapitre Gradient projeté perturbé.

6.6.2 Résultats numériques

Dans le tableaux suivant les résultats numériques des problèmes en utilisant le recuit simulé.
PN est le numéro du problème, x^* le vecteur optimal et $f(x^*)$ la valeur optimal de la fonction objectif.
Tous les programmes ont été implémenté en Matlab 6.5.1, sous Windows XP et Pentium (R) 4 CPU 1400 MHZ.

PN	x^*	f(x^*)	temps(sec)
1	$(6,\ 1.14,\ 0.13,\ 0,\ 7.71,\ 0,\ 10.63,\ 0)$	94.58	14.05
2	$(81.51,\ 42.17,\ 36.86,\ 44.69,\ 27.65)$	12201.87	1.22
3	$(42.28,\ 44.07,\ 69.11,\ 0.05)$	465639.81	5.29
4	$(-0.0071,\ -0.0405)$	0.0014	0.18
5	$(1,\ 0.18,\ -0.98)$	-10.36	0.37
6	$(-0.01,\ -0.75)$	-0.99	0.15

TABLE 6.1 – Résultats numériques du recuit simulé

Interprétation des résultats

On remarque que le recuit simulé donne parfois de bons résultats, et qu'il s'approche vers l'optimum et ceci est dû au remontées qu'il accepte de faire avec une probabilité donnée.

6.7 Conclusion

Le recuit simulé a montré son efficacité pour les problèmes NP, comme le problème du voyageur de commerce discuté dans ce chapitre, et on a pû l'utiliser aussi pour des problèmes d'optimisation continus.
Cependant, son principal inconvénient réside dans le choix de 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.
Le couplage du recuit simulé avec les algorithmes déterministes rend ces méthodes plus robustes et moins dépendantes du choix des paramètres, ce qui est

un avantage indéniable. En autre, on constate une amélioration de la qualité des résultats.

Chapitre 7

Hybridation

7.1 Introduction

La non convergence vers un optimum global lors de l'utilisation d'algorithmes déterministes défavorise leur affectation lors de problèmes d'optimisation non convexe.
En effet, l'algorithme de Piyavskii est très lent au voisinage de l'optimum.
L'intérêt de la perturbation aléatoire de l'algorithme déterministe, comme par exemple, dans notre cas : l'algorithme du gradient projeté et de la méthode SQP est d'éviter les optimums locaux.
Cependant, cette perturbation n'est pas toujours satisfaisantes.
Les algorithmes stochastiques comme :

- l'algorithme génétique
- le recuit simulé

prouvent leur robustesse dans certains cas ; notamment où la fonction objectif n'est pas dérivable, ou bien la petitesse de l'intervale de recherche.
Ainsi, dans ce chapitre on va coupler l'algorithme de Piyavskii avec l'algorithme génétique, l'algorithme du gradient projeté perturbé et la méthode SQP perturbée avec l'algorithme génétique et le gradient projeté perturbé avec le recuit simulé.
La similarité des problèmes tests classiques de l'ensemble des chapitres, est une preuve, concrète, de l'efficience de l'hybridation.

Notre principal argumentaire de démonstration s'axe autour de deux points :

- le temps de calcul est inférieur
- le nombre d'évaluation de la fonction objectif et des contraintes est aussi inférieur
- l'optimum obtenu est meilleur.

Il en découle que l'hybridation est un processus détérminant dans l'optique de l'amélioration et de la mise en valeur des algorithmes d'optimisation.

7.2 Couplage de l'algorithme de Piyavskii et de l'algorithme génétique

7.2.1 Méthodologie

Soit le problème suivant :

$$x^* = \operatorname{argmin} \{J(x) : x \in C\} \tag{7.1}$$

où $J : I\!R^{n-1} \longrightarrow I\!R$ est une fonction régulière et Lipshitzienne qui n'est pas forcément convexe ou linéaire et C un ensemble régulier non vide. C est définie par :

$$C = \{x \in I\!R^{n-1} \,/\, h_i(x) \leq 0,\, i = 1, \ldots, p \quad \text{et} \quad h_i(x) = 0,\, i = p+1, \ldots, q\}. \tag{7.2}$$

On a vu dans le chapitre 2, que l'algorithme de Piyavskii est convergent pour ce type de problèmes et donne un optimum global.
Puisque l'algoritme de Piyavskii est lent au voisinage de l'optimum global et l'algorithme génétique est plus robuste au niveau des petits intervalles, car il peut trouver l'optimum global facilement en effectuant les opérateurs d'évolution.
L'idée est d'appliquer l'algorithme de Piyavskii jusqu'à une précision sans pour autant s'approcher de l'optimum, puis utiliser l'algorithme génétique.
En effet, l'algorithme génétique donne un vecteur optimal (x^*) et la valeur de la fonction objectif en ce point.
L'algorithme génétique qu'on a implémenté opère sur un fonction définie sur un intervalle.

L'idée du couplage vient du fait que l'algorithme de Piyavskii est très lent au voisinage de l'optimum mais aussi très rapide quand il est loin de la solution. De plus, il a été prouvé qu'il trouve l'optimum global.
L'algorithme génétique que nous avons programmé opère avec un nombre constant d'itération.
Donc si l'intervalle de recherche est grand il peut ne pas converger et donner des solutions lointaines de l'optimum global. Cependant, si l'intervalle de recherche est petit, les chances de tomber sur l'optimum global ou au d'en être proche sont maximales.
L'avantage de cet algorithme est qu'il opère avec un nombre constant d'itérations.
Le couplage de l'algorithme de Piyavskii avec l'algorithme génétique est implémenté comme suit :

- On applique l'algorithme de Piyavskii jusqu'à une précision donnée pas très grande (on a choisi la précision égale à $\eta = 1.0, 0.5, 0.25, \ldots$) ce qui va nous donner un vecteur x.
- Après, on applique l'algorithme génétique à la fonction sur l'intervalle centré en x et de longueur η (c'est un intervalle relativement petit), en forçant l'algorithme génétique à mettre x dans la population initiale.

Pour les problèmes avec contraintes, l'idée du couplage est d'implémenter à nouveaux l'algorithme de Piyavskii jusqu'à une précision donnée η. Ce qui va donner un vecteur optimal x^*.
Par la suite, on applique l'algorithme génétique sur l'intervalle centré en x^* et de longueur η.
Le vecteur trouvé par l'algorithme génétique sera corrigé en utilisant la méthode de correction 3.30 définie dans le chapitre 3. Ladite méthode consiste à ramener un point irréalisable vers la frontière. Dans le cas échéant, la méthode de correction sera efficace car le point trouvé ne sera pas très loin de la frontière, du fait de la longueur pas très grande de l'intervalle η.

7.2.2 Résultats numériques

Problèmes sans contraintes

Les problèmes sans contraintes utilisés pour l'évaluation du couplage de l'algorithme génétique avec l'algorithme de Piyavskii sont ceux définit dans la table 2.4.1 du chapitre 2.

Problèmes avec contraintes

Les problèmes avec contraintes utilisés pour l'évaluation du couplage de l'algorithme génétique avec l'algorithme de Piyavskii sont les problèmes 1-4 du chapitre 2.

Résultats numériques

Dans les tableaux suivants les résultats numériques des problèmes définie ci-dessus. Tous les programmes ont été implémenté en Matlab 6.5.1, sous Windows XP et Pentium (R) 4 CPU 1400 MHZ.

Numéro de la fonction	Valeur optimale	Vecteur optimal	Nombre d'évaluation	temps (sec)
1	2.5198	(0.9999,0.6351)	11256	0.12
2	0.9999	(1.5703,1.0008)	653	3.35
3	0.9999	(1.5711,-1.5706)	929	2.32
4	2662.3528	(-0.7736,-0.7736,-1.4320)	28760	10.01
5	0.9999	(0.4003,-0.5326)	299	16.14
6	1.9126	(-0.5625,-1.5471)	1738	16.09
7	-0.1692	(1.7938,1.3779)	1652	16.15
8	-1.7441	(1.7410,2.0299)	9260	16.96

TABLE 7.1 – Résultats numériques du couplage de l'algorithme de Piyavskii et l'algorithme génétique sans contraintes

Numéro de la fonction	valeur f_{opt}	vecteur optimal	temps (sec)
1	2688.7850	(-2.1808,2.9388,2.0449,-0.2914,-1.0325)	1.70
2	-4.5955	(1.4012,1.8098,1.8037)	26.21
3	28.6419	(9.4444,0.8333)	$1.56.10^{-2}$
4	6.7308	(1,1.0740,0.7777,1.8888)	1.01

TABLE 7.2 – Résultats numériques du couplage de l'algorithme de Piyavskii et de l'algorithme génétique avec contraintes

7.2.3 Récapitulation

Problèmes sans contraintes

Dans le tableau qui suit les résultats des 8 problèmes sans contraintes obtenus en utilisant les 3 algorithmes : les 3 premières colonnes représentent l'algorithme de Piyavskii, les 3 deuxième colonnes l'algorithme génétique et les 3 dernières colonnes le couplage de ces deux algorithmes.

Numéro de la fonction	Valeur optimale	temps CPU	Eval	Valeur optimale	temps CPU	Eval	Valeur optimale	temps CPU	Eval
1	2.5177	16.31	12901	2.5083	16.20	100	2.5198	0.12	11256
2	0.9990	16.01	44275	0.9999	16.34	100	0.9999	3.35	653
3	0.9999	16.01	19120	0.9999	16.10	100	0.9999	2.32	929
4	2662.3528	26.95	28660	50.89	24.10	100	2662.3528	10.01	28760
5	0.9999	53.51	42154	0.9999	15.96	100	0.9999	16.14	299
6	1.9132	30.50	43927	-1.2284	16.43	100	1.9126	16.09	1738
7	-0.1690	70.03	57649	-0.1691	16.28	100	-0.1692	16.15	1652
8	-1.7441	33.67	26674	-1.7441	16.23	100	-1.7441	16.96	9260

TABLE 7.3 – Résultats des problèmes sans contraintes

7.2.4 Problèmes avec contraintes

Dans le tableau qui suit les résultats des 4 problèmes avec contraintes obtenus en utilisant les 3 algorithmes : les 3 premières colonnes représentent l'algorithme de Piyavskii, les 3 deuxième colonnes l'algorithme génétique et les 3 dernières colonnes le couplage de ces deux algorithmes.

Numéro de la fonction	Valeur optimale	temps CPU	Eval	Valeur optimale	temps CPU	Eval	Valeur optimale	temps CPU	Eval
1	2411.3513	40.71	1855	2819	42.87	100	2688.78	1.70	1322
2	-4.5955	38.23	17533	-4.6839	26.31	100	-4.60	26.21	2589
3	28.6419	27.35	123	n.cvt	22.2	100	28.65	$1.56.10^{-2}$	46
4	6.2331	40.79	2026	n.cvt	23.35	100	6.2331	1.01	1263

TABLE 7.4 – Résultats des problèmes avec contraintes

7.2.5 Interprétation des résultats

A travers la comparaison illustrée dans les tableaux ci-dessus, on peut voir clairement que le couplage de l'algorithme de Piyavskii et l'algorithme génétique génére un optimum meilleur que les deux algorithmes tout en minimisant remarquablement le nombre d'évaluation de la fonction et le temps de calcul.

7.2.6 Conclusion

Le couplage de l'algorithme de Piyavskii avec l'algorithme génétique a fourni de bons résultats.
La précision de l'algorithme de Piyavskii a été gardée tout en améliorant le temps de calcul et le nombre d'évaluation de la fonction.
Néanmoins ce couplage reste limité puisqu'il opère seulement sur les fonctions dites de Lipshitz.

7.3 Couplage du gradient projeté perturbé et de l'algorithme génétique

7.3.1 Méthodologie

Théoriquement le gradient projeté perturbé converge vers un optimum global.
En pratique, il peut être piégé par un optimum local, en fonction des choix des valeurs des paramètres.
Cette constatation nous a poussé à rendre l'algorithme du gradient projeté perturbé sous une forme hybride.

Cette approche hybride a l'avantage de ne pas dépendre de ces paramètres et en plus générer plus de solutions au lieu d'une.
En plus elle n'a pas besoin d'un point initial.
Dans ce qui suit, on considère un couplage entre le gradient projeté perturbé définie dans le chapitre 3 et une méthode évolutionnaire.
Initialement, on génére une population $\mathbf{S}_0 = \{x_0^1, \ldots, x_0^N\}$ de points réalisables comme suit ;

- On génére un population aléatoire de points $\{xt_0^1, \ldots, xt_0^N\}$ dans l'intervalle de recherche.
- Chaque point est ramené à la frontière, pour être un point réalisable, en utilisant la méthode de correction 3.30 décrite dans le chapitre 3.

Les itérations génèrent $S_1,\ S_2,\ \ldots$ comme suit :
à l'itération k, $\mathbf{S}_k = \{x_k^1, \ldots, x_k^N\}$ est donné et on génére $\mathbf{S}_{k+1} = \{x_{k+1}^1, \ldots, x_{k+1}^N\}$ en utilisant 3 opérateurs : **R** (croisement), **M** (mutation), **S** (sélection). L'application de ce ces opérateurs est faite de la manière suivante :

- On applique l'opérateur de croisement à la population : On génére r éléments qui ont la forme $\alpha x_k^i + \beta x_k^j + \gamma$, où α, β et γ sont choisis aléatoirement. Ensuite on corrige ces éléments en utilisant la méthode de correction 3.30 du chapitre 3, pour avoir un ensemble réalisable $\mathbf{R}_k$.
- On applique l'opérateur de mutation à la population : on génére t éléments en utilisant le gradient projeté perturbé définie dans le chapitre 3 appliqué à tous les éléments de $\mathbf{S}_k$, On aura l'ensemble alors $\mathbf{M}_k$.
- On applique l'opérateur de sélection à la population : on choisie les meilleurs N éléments de l'ensemble $\mathbf{A}_k = \mathbf{S}_k \cup \mathbf{R}_k \cup \mathbf{M}_k$, ce qui va donner un nouveau ensemble réalisable

$$\mathbf{S}_{k+1} = \{X_{k+1}^1, \ldots, X_{k+1}^N\}.$$

Les itérations sont stoppés si $\| \mathbf{S}_{k+1} - \mathbf{S}_k \| \leq \epsilon$, où $\epsilon << 1$.

7.3.2 Résultats numériques

Dans cette section on va donner des résultats numériques de 8 problèmes implémentés par l'algorithme hybride.

Définition des problèmes

Les problèmes 1-6 utilisés pour l'évaluation du couplage de l'algorithme génétique avec le gradient projeté perturbé sont les problèmes 1-6 du chapitre 3.

Problème 7

La source de ce problème est [52]. La dimension est $n = 100$ et la fonction objectif est :

$$J(x) = 1 + 6\left(x_1 - \sum_{i=2}^{100} x_i\right)^2 - \cos\left(12\left(x_1 - \sum_{i=2}^{100} x_i\right)\right)$$
$$+1000\sum_{i=2}^{100} x_i^2$$

Le problème a 2 contraintes données par :

$$\sum_{i=6}^{100} x_i^2 - x_5^2 = 0;$$
$$x_2^2 + x_4^2 - x_3^2 = 0;$$

Théoriquement la solution de J est :
$J(x^*) = 0$ et $x^* = 0$

Problème 8

La source de ce problème est [52]. La dimension est $n = 100$ et la fonction objectif est :

$$J(x) = 2 + 6\left(x_1 - \sin\left(\sum_{i=3}^{100} x_i\right)\right)^2$$
$$-\cos\left(12\left(x_1 - \sin\left(\sum_{i=3}^{100} x_i\right)\right)\right)$$
$$+1000\sum_{i=3}^{100} x_i^2 + 6\left(x_2 - \sin\left(\sum_{i=3}^{100} x_i\right)\right)^2$$

$$-\cos\left(12\left(x_2-\sin\left(\sum_{i=3}^{100}x_i\right)\right)\right)$$

Le problème a 3 contraintes données par :

$$\sum_{i=8}^{98}x_i^2-x_6^2=0;$$
$$x_{99}^2+x_{100}^2=0;$$
$$x_7^2+x_3^2+x_5^2-x_4^2=0;$$

Théoriquement la solution de J est :
$J(x^*)=0$ et $x^*=0$

Résultats numériques

Dans les tableaux suivants les résultats numériques des problèmes définie ci-dessus. Tous les programmes ont été implémenté en Matlab 6.5.1, sous Windows XP et Pentium (R) 4 CPU 1400 MHZ.

PN	x^*	J(x^*)	temps
1	$(1.02\times10^{-18},\ 2.37,\ 1.00,\ 5.38,\ 6.75,\ 0.78,\ 1.55,\ 1.54)$	35.12	5.2
2	$(78,\ 33,\ 29.9969,\ 45,\ 36.7720)$	10122.65789	10.23
3	$(42.2866,\ 44.0838,\ 70,\ 0.5253)$	486552.9507	30.15
4	$(-0.0001,\ 0.0009)$	9.4643×10^{-07}	42.01
5	$(-2.78,\ -2.57,\ 1.33,\ 1.06,\ 1.86)$	-45.03	5.24
6	$(-0.08995609,\ -0.71327529)$	-1.03162533	5.67
7	Approximativement 10^{-4}	0.0024	50.22
8	Approximativement 10^{-5}	0.000036	50.11

TABLE 7.5 – Résultats numériques du couplage du gradient projeté perturbé et de l'algorithme génétique

7.3.3 Récapitulation

Dans le tableau qui suit les résultats des 6 premiers problèmes obtenus en utilisant les 3 algorithmes : les 2 premières colonnes représentent l'algorithme du gradient projeté perturbé, les 2 deuxième colonnes l'algorithme génétique et les 2 dernières colonnes le couplage de ces deux algorithmes.

Numéro de la fonction	Valeur optimale	temps CPU	Valeur optimale	temps CPU	Valeur optimale	temps CPU
1	35.12	5.34	n.cvt	23.35	35.12	5.2
2	10123	13	n.cvt	23.35	10122.65	10.23
3	538510	19.22	n.cvt	23.35	486552.9	30.15
4	0.6617	0.78	n.cvt	23.35	$9.46.10^{-07}$	42.01
5	-12.17	1.03	n.cvt	23.35	-45.03	5.24
6	-0.2152	0.18	n.cvt	23.35	-1.03	5.67

TABLE 7.6 – Comparaison des résultats des problèmes

7.3.4 Interprétation des résultats

La comparaison des 3 algorithmes confirme l'efficacité du couplage des deux algorithmes au niveau de la précision de l'optimum.

7.3.5 Conclusion

La satisfaction des résultats obtenus confère à l'hybridation sa crédibilité. En fait, en utilisant cette approche on a pû éviter les optimums locaux profonds et aussi le choix du point initial.
Même s'il y'a une petite augmentation du temps du calcul dûe à l'application de plusieurs algorithmes simultanément, on peut dire que cette approche hybride est performante et est meilleure que l'algorithme du gradient projeté perturbé.
Nous avons aussi couplé la méthode SQP perturbée et l'algorithme génétique en respectant la même méthodologie. Il s'est avéré que nous avons trouvé les mêmes résultats.

7.4 Couplage du gradient projeté perturbé et du récuit simulé

Dans cette section, on va présenter un autre algorithme hybride construit à partir du couplage de l'algorithme du gradient projeté perturbé et du récuit simulé.

7.4.1 Méthodologie

Comme on a vu dans le chapitre 3, à chaque itération de l'algorithme du gradient projeté perturbé on trouve un nouveau vecteur X_{k+1} qui est obtenu par la correction, en utilisant la méthode 3.30, du vecteur XT_{k+1} obtenu par la transformation

$$XT_{k+1} = X_k + \eta_k D_k,$$

où η_k est donnée par 3.29 et D_k est la direction de descente perturbée definie par

$$D_k = d_k + \xi_k Z_k.$$

Comme on l'a mentioné dans la section précédente, l'algorithme du gradient projeté perturbé peut ne pas converger vers un optimum global.
L'idée du couplage du gradient projeté perturbé et le récuit simulé est la suivante, à chaque itération du gradient projeté perturbé, on applique le récuit simulé en prenant comme point initial du récuit le point XT_{k+1}, on va trouver alors un point YT_{k+1}, puis on corrige ce point en utilisant la méthode de correction 3.30, ce qui va donner un point Y_{k+1} qui est réalisable pour l'algorithme du gradient projeté perturbé. Si on note $\{WW_k\}_{k\geq 0}$ la suite générée par l'algorithme hybride, on aura deux cas :

- si le point obtenu par le récuit simulé est mieux que X_{k+1}, on pose $WW_{k+1} = Y_{k+1}$
- sinon on implémente normalement le gradient projeté perturbé, et $WW_{k+1} = X_{k+1}$

L'avantage de cet algorithme est qu'il peut éviter n'importe quel optimum local, et le nombre d'évaluation de la fonction objectif et des contraintes est moins que si on applique l'algorithme du gradient projeté perturbé, et l'optimum est mieux que si on applique le récuit simulé.

7.4.2 Théorème de convergence

Il est clair que la suite $W_k = J(WW_k)$ est décroissante et bornée inférieurement par α_0. Donc, il existe $W \geq \alpha_0$ tel que $W_k \longrightarrow W$ quand $k \longrightarrow +\infty$.
Avec les mêmes hypothèses que l'algorithme du gradient projeté perturbé, l'algorithme hybride converge vers un optimum global :

Théorème 7.1 *Supposons que $W_0 = X_0 \in S$, 3.38 est satisfaite et*

$$\sum_{k=0}^{+\infty} \frac{1}{\xi_k} = +\infty \tag{7.3}$$

alors $W = \alpha_0$ presque surêment.

Démonstration

On a

$$\forall k \qquad W_k \leq U_k$$

où $U_k = J(X_k)$ est la suite générée par le gradient projeté perturbé.
Donc

$$P(W_k \geq \theta) \leq P(U_k \geq \theta)$$

et on aura

$$P(W_{k+1} \geq \theta \mid W_k \geq \theta) \leq P(U_{k+1} \geq \theta \mid U_k \geq \theta) \tag{7.4}$$

On a aussi

$$P(U_{k+1} < \theta \mid U_k \geq \theta) = 1 - P(U_{k+1} \geq \theta \mid U_k \geq \theta) \tag{7.5}$$

$$P(W_{k+1} < \theta \mid W_k \geq \theta) = 1 - P(W_{k+1} \geq \theta \mid W_k \geq \theta) \tag{7.6}$$

De 7.4 on a

$$1 - P(U_{k+1} \geq \theta \mid U_k \geq \theta) \leq 1 - P(W_{k+1} \geq \theta \mid W_k \geq \theta)$$

alors

$$P(U_{k+1} < \theta \mid U_k \geq \theta) \leq P(W_{k+1} < \theta \mid W_k \geq \theta) \tag{7.7}$$

Et

$$P(U_{k+1} < \theta \mid U_k \geq \theta) = P(X_{k+1} \in S_\theta \mid X_k \notin S_\theta)$$

on a $\{X_k\}_{k\geq 0} \subset S$, donc [10] :

$$P(X_{k+1} \in S_\theta \ , \ X_k \notin S_\theta) = \int_{S - S_\theta} P(X_k \in dx) \int_{S_\theta} f_{k+1}(y \mid X_k = x) dy.$$

Puisque

$$P(X_{k+1} \in S_\theta \mid X_k \notin S_\theta) = \frac{P(X_{k+1} \in S_\theta \ X_k \notin S_\theta)}{P(X_k \notin S_\theta)},$$

on a

$$P(X_{k+1} \in S_\theta \mid X_k \notin S_\theta) \geq \inf_{x \in S - S_\theta} \left\{ \int_{S_\theta} f_{k+1}(y \mid X_k = x) dy \right\}$$

On a aussi

$$\begin{array}{rcl} P(X_{k+1} \in S_\theta \mid X_k \notin S_\theta) & \geq & \frac{1}{\eta_k^n \xi_k^n} \inf_{x \in S - S_\theta} \left\{ \int_{S_\theta} \phi_k \left(\frac{y - Q_k(x)}{\eta_k \xi_k} \right) dy \right\} \\ & \geq & \frac{\mathsf{meas}(S_\theta)}{\eta_k^n \xi_k^n} \inf_{x \in S - S_\theta, y \in S_\theta} \left\{ \phi_k \left(\frac{y - Q_k(x)}{\eta_k \xi_k} \right) \right\} . \end{array}$$

En utilisant la proposition 3.2, on a

$$P(X_{k+1} \in S_\theta \mid X_k \notin S_\theta) \geq \frac{\mathsf{meas}(S_\theta)}{\xi_k^n} \left(\frac{6\lambda}{\alpha} \right)^n \psi \left(\frac{M}{\eta_k \xi_k} \right)$$

et

$$\psi \left(\frac{M}{\eta_k \xi_k} \right) \geq \psi \left(\frac{M}{16 \alpha \delta \xi_k} \right) \geq \psi \left(\frac{M}{16 \alpha \delta \xi_0} \right)$$

Donc,

$$P(X_{k+1} \in S_\theta \mid X_k \notin S_\theta) \geq c_\theta(k) \tag{7.8}$$

où

$$c_\theta(k) = \frac{\gamma(\theta)}{\xi_k^n} \; ; \; \gamma(\theta) = \mathsf{meas}(S_\theta) \left(\frac{6\lambda}{\alpha} \right)^n \psi \left(\frac{M}{16 \alpha \delta \xi_0} \right)$$

De 7.7 on a

$$P(W_{k+1} < \theta \mid W_k \geq \theta) \geq c_\theta(k)$$

on a alors

$$\forall \theta \in]\alpha_0, \alpha_1[: \, c_\theta(k) > 0 \, .$$

L'équation 7.3 donne

$$\sum_{k=0}^{+\infty} c_\theta(k) = +\infty \, .$$

En appliquant le lemme 3.1 on a : $W = \alpha_0$ sûrement.

7.4.3 Résultats numériques

Dans cette section on va donner des résultats numériques de 6 problèmes implémentés par l'algorithme hybride.

Définition des problèmes

Les problèmes utilisés pour l'évaluation du couplage du couplage du gradient projeté perturbé et du recuit simulé sont les problèmes 1-6 de la section précédente.

Résultats numériques

Dans les tableaux suivants les résultats numériques des problèmes définis ci-dessus. Tous les programmes ont été implémenté en Matlab 6.5.1, sous Windows XP et Pentium (R) 4 CPU 1400 MHZ.

PN	x^*	J(x^*)	temps
1	(0, 3.59, 1.32, 4.85, 4.88, 0.70, 1.52, 0.97)	40.21	2.67
2	$(88.62,\ 34.27,\ 44.31,\ 27.08,\ 40.94)$	10587.51	11.35
3	$(44,\ 40,\ 60,\ 0.14)$	440507.26	31.25
4	$(0.0032,\ -0.0123)$	2.11×10^{-04}	0.28
5	$(1,\ -0.19,\ -0.98,\ -6.04)$	-8.81	4.23
6	$(0.10,\ 0.72)$	-1.02	0.27

TABLE 7.7 – Résultats numériques du couplage du gradient projeté perturbé et du récuit simulé

7.4.4 Récapitulation

Dans le tableau qui suit les résultats des 6 premiers problèmes obtenus en utilisant les 3 algorithmes : les 2 premières colonnes représentent la méthode SQP perturbée, les 2 deuxième colonnes le recuit simulé et les 2 dernières colonnes le couplage de ces deux algorithmes.

7.4.5 Interprétation des résultats

La comparaison des 3 algorithmes confirme l'efficacité du couplage des deux algorithmes dans la plupart des cas, et la cause qui engendre les cas où le couplage ne donne pas de bons résultats par rapport aux deux autres algorithmes,

Numéro de la fonction	Valeur optimale	temps CPU	Valeur optimale	temps CPU	Valeur optimale	temps CPU
1	35.09	1.04	94.58	14.05	40.21	2.67
2	10484	0.64	12201	1.22	10587.51	11.35
3	538510	6.22	465639.81	5.29	440507.26	31.25
4	0.6617	0.67	0.0014	0.18	2.11×10^{-04}	0.28
5	-12.17	0.86	-10.36	0.37	-8.81	4.23
6	-0.2152	0.18	-0.99	0.15	-1.02	0.27

TABLE 7.8 – Comparaison des résultats des problèmes

est que le recuit accepte parfois des remontées, on peut conclure alors que le couplage esst efficace mais dans des cas seulement.

7.4.6 Conclusion

Les résultats fournis par le couplage du gradient projeté perturbé et du récuit simulé tend à prouver l'efficacité de la méthodologie proposée.
L'hybridation donne des résultats plus intéressants au niveau de la precision de l'optimum ainsi que le temps de calcul et le nombre d'évaluation de la fonction objectif et des contraintes.

Troisième partie

Application

Chapitre 8

Un problème logistique

8.1 Introduction

Pour montrer l'efficacité de l'hybridation, on a choisi d'appliquer le couplage du gradient projeté perturbé et l'algorithme génétique discuté dans le chapitre 7 à un problème de transport.
Etant donné un dépôt central et plusieurs points de demandes dans une région $\mathcal{R}$, le problème de transport choisi consiste à désigner un ensembles de trajets depuis le dépôt aux points de demandes tout en minimisant le coût [4].
La région $\mathcal{R}$ est divisée en districts, chaque district est assigné à un véhicule. Chaque véhicule démarre du dépôt vers le district approprié, délivre la marchandise à tous les points de demandes dans ce district et revient vers le dépôt. Dans ce chapitre, on veut trouver le partitionnement optimal de $\mathcal{R}$ basé sur un système de coordonnées polaires centré au dépôt en districts tel que

- La capacité maximale du véhicule est fixée.
- Le temps de voyage d'un véhicule par jour est fixé.
- Le partitionnement doit guarantir un certaine homogénité entre les districts.

8.2 Le modèle

8.2.1 Généralitées

Le véhicule part au district de surface A, délivre la marchandise à n points de demande. La distance totale $E(DZ)$ parcourue par le véhicule peut être approchée par [39] :

$$E(DZ) = k_0 \sqrt{An}. \tag{8.1}$$

où k_0 est une constante qui dépend de la région [31].
Un véhicule démarre du dépôt vers le district assigné, délivre la marchandise à tous les points de demande dans ce district ou quand le temps de travail maximal est atteint, et il revient au dépôt. Cette suite d'étapes est appelée le cycle du véhicule. L'espérance $\overline{T}$ du temps total du cycle T est donnée par [40] :

$$\overline{T} = E(T) = \frac{2\,E(D_{LH})}{v_L} + \frac{k_0\sqrt{An}}{v_Z} + p\,n\,E(t_S). \tag{8.2}$$

Où $E(D_{LH})$ est l'espérance de la distance depuis le dépôt au district, v_L est la vitesse moyenne entre le dépôt et le district, v_Z est la vitesse moyenne dans le district, $E(t_S)$ est l'espérance du temps d'arrêt dans une livraison et p la probabilité qu'un client soit visité.

Soit t_h le temps du voyage entre le dépôt et le district, et t_Z le temps du voyage dans le district, la variance de T est

$$var(T) = \sigma_T^2 = 2\,var(t_h) + var(t_Z) + p\,n\,var(t_S). \tag{8.3}$$

On note $E(u)$, σ_u et U respectivement la moyenne, la déviation standart de la quantité du produit u livré par point visité dans le district et le poids total du véhicule, l'espérance de U est

$$\overline{U} = E(U) = p\,n\,E(u). \tag{8.4}$$

La variance de U peut être écrite

$$var(U) = \sigma_U^2 = p\,n\,\sigma_u^2. \tag{8.5}$$

T et U peuvent être représentés respectivement par les distributions normales $N(\overline{T}, \sigma_T)$ et $N(\overline{U}, \sigma_U)$. Soit $\zeta \sim N(0,1)$ la variété unité normal. En adoptant un

98 pourcentage, $\zeta = 2.06$, les restrictions suivantes doivent être respéctées :

$$g_1 = \overline{T} + 2.06\,\sigma_T \leq H_1. \tag{8.6}$$

Et

$$g_2 = \overline{U} + 2.06\,\sigma_U \leq W. \tag{8.7}$$

Où H_1 est le nombre d'heures par jour que le temps du cycle ne peut pas dépasser, et W est la capacité du véhicule.

8.2.2 Le modèle

Le partitionnement de la région est basé sur un système de coordonnées polaires. Soit $Z_{i,j}$ un district limité par les rayons r_{i-1} et r_i, et les angles $\theta_{i-1,j}$ et $\theta_{i,j}$, la surface du district est alors

$$A_{i,j} = \frac{\theta_{i,j} - \theta_{i,j-1}}{2}\,(r_i^2 - r_{i-1}^2). \tag{8.8}$$

Le nombre de points visités dans un district $Z_{i,j}$ est [40] :

$$n_{i,j} = N(r_i, \theta_{i,j}) - N(r_i, \theta_{i,j-1}) - N(r_{i-1}, \theta_{i,j}) + N(r_{i-1}, \theta_{i,j-1}). \tag{8.9}$$

Où $N(r, \theta)$ est la fonction cumulative des points visités dans le secteur $r,\ \theta$. Le temps total dépensé dans un district est [40] :

$$\gamma_{i,j} = \psi(r_i, \theta_{i,j}) - \psi(r_i, \theta_{i,j-1}) - \psi(r_{i-1}, \theta_{i,j}) + \psi(r_{i-1}, \theta_{i,j-1}). \tag{8.10}$$

Où $\psi(r, \theta)$ est la fonction cumulative du temps dépensé dans la livraisons aux points visités.

On définie une *fonction de distribution d'efforts* par

$$E_{i,j} = \frac{\frac{k_0\sqrt{A_{i,j}n_{i,j}}}{v_Z} + p\,\gamma_{i,j}}{f_{i,j}\,H_1}, \tag{8.11}$$

où

$$f_{i,j} = \frac{1 - 2\,r_{i-1}/v_L}{H_1}$$

Les districts $Z_{i,j}, \quad j = 1, 2, \ldots$ sont contenus dans un *anneau* i, soit $\overline{E}_i$ la moyenne de l'effort dans l'anneau i. Pour guarantir l'homogénité entre les districts, la contrainte suivante doit être respectée [40] :

$$\sum_{i=1}^{NR} \sum_{j=1}^{ND_i} \frac{E_{i,j} - \overline{E}_i}{\overline{E}_i} \leq 0. \tag{8.12}$$

Où NR est le nombre d'anneaux dans la région, et ND_i le nombre de districts dans l'anneau i. En appliquant les restrictions 8.6 et 8.7 à chaque distict, on aurra :

$$\sum_{i=1}^{NR} \sum_{j=1}^{ND_i} g_1 - H1 \leq 0. \tag{8.13}$$

$$\sum_{i=1}^{NR} \sum_{j=1}^{ND_i} g_2 - W \leq 0. \tag{8.14}$$

La fonction objectif s'écrit [40] :

$$\sum_{i=1}^{NR} \sum_{j=1}^{ND_i} C_{Km} E(D_{i,j}) + C_H E(T_{i,j}) + C_{Ov} H X_{i,j}. \tag{8.15}$$

Où C_{Km} est le coût par kilomètre, C_H le coût par heure, C_{Ov} le coût de travail supplémentaire calculé quand le temps du cycle de travaille dépasse une limite H_0 et $HX_{i,j}$ l'espérance du temps de travail supplémentaire par jour dans le district $Z_{i,j}$.

8.2.3 Résultats numériques

La région $\mathcal{R}$ est divisée en districts en utilisant un système de coordonnées polaires centré au dépôt, les angles varient entre $\pi/100$ et π, et le rayon est $60\,Km$

nr est le nombre d'anneaux dans $\mathcal{R}$, le nombre de districts est $nr * nt$, et n est le nombre de points visités par région.

La vitesse moyenne est $v_L = 35\,Km/h$ et la vitesse dans un district est $v_Z = 24\,Km/h$.

La quantité du produit livrée à chaque client est $5.76\,Kg$ avec une déviation standard de $2.08\,Kg$. le temps d'arrêt est $3\,min\,49s$.

Le temps normal de travail est $H_0 = 8\,hrs/jour$, et au maximum $H_1 = 10\,hrs/jour$. La capacité des véhicules est $w = 500\,Kg$, la constante k_0 définie dans (8.1), peut être approchée par $k_0 = k_1 k_2$ [31], où k_1 dépend de la métrique adoptée et k_2 dépend de la structure de la route, on prend $k_1 = 0.765$ and $k_2 = 1.35$.
Les coût sont :

$$C_{Km} = a_0 w^{a_1}, \quad C_H = b_0 w^{b_1}, \quad C_{Ov} = C_H + 8. \tag{8.16}$$

où : $a_0 = 0.00408, \quad a_1 = 0.5, \quad b_0 = 0.48, \quad b_1 = 0.4, \quad w = 500.$

On a implémenté le modèle sur MATLAB. On prend $n = 2500$ et pour bien comparer les résultats, les points visités sont distribués uniformément dans la région.
En parallélle, nous appliquons le programme à $n = 30$ point aléatoirement répartis dans la région.
En changeant le nombre de rayons nr et le nombres d'angles nt, on obtient les résultats suivants :

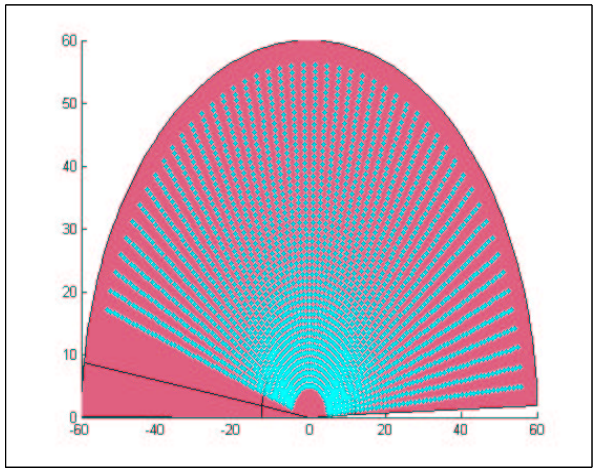

FIGURE 8.1 – Résultats pour le cas uniforme : nr = 3 et nt = 3

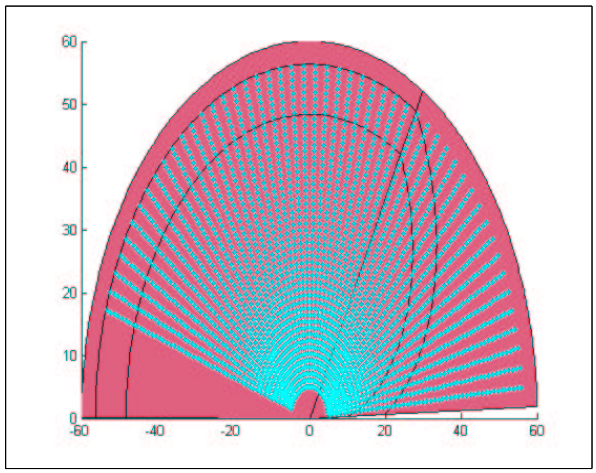

FIGURE 8.2 – Résultats pour le cas uniforme : nr = 4 et nt = 3

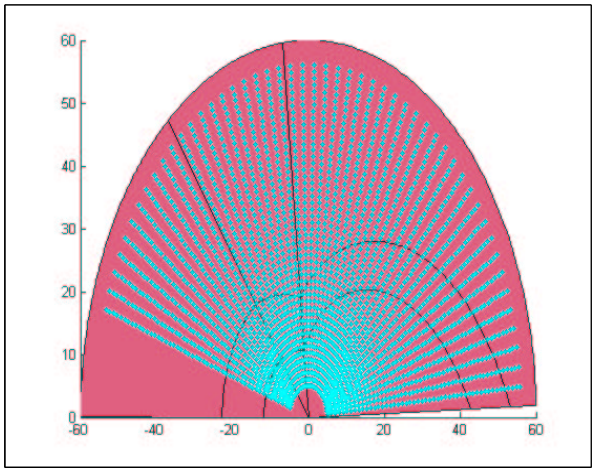

FIGURE 8.3 – Résultats pour le cas uniforme : nr = 4 et nt = 4

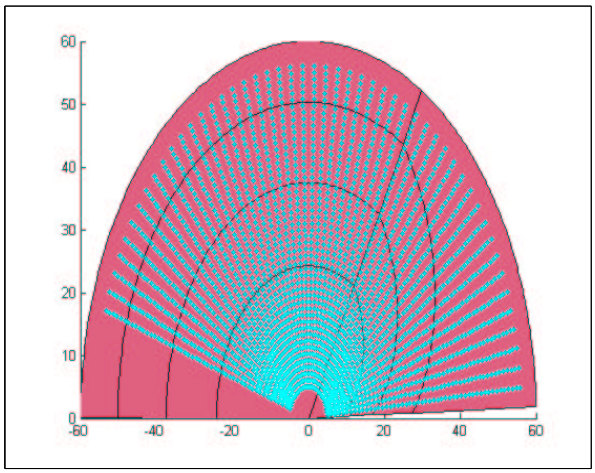

FIGURE 8.4 – Résultats pour le cas uniforme : nr = 5 et nt = 3

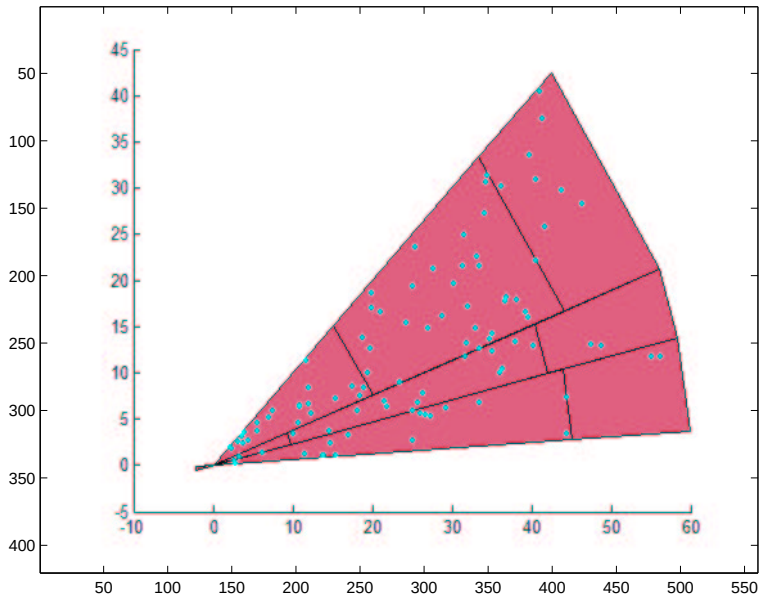

FIGURE 8.5 – Résultats pour le cas non uniforme : nr = 3 et nt = 4

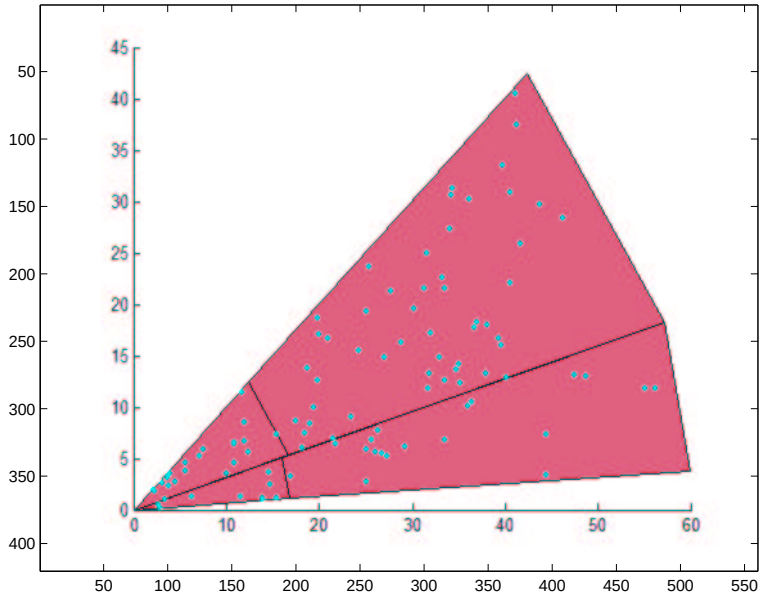

FIGURE 8.6 – Résultats pour le cas non uniforme : nr = 3 et nt = 3

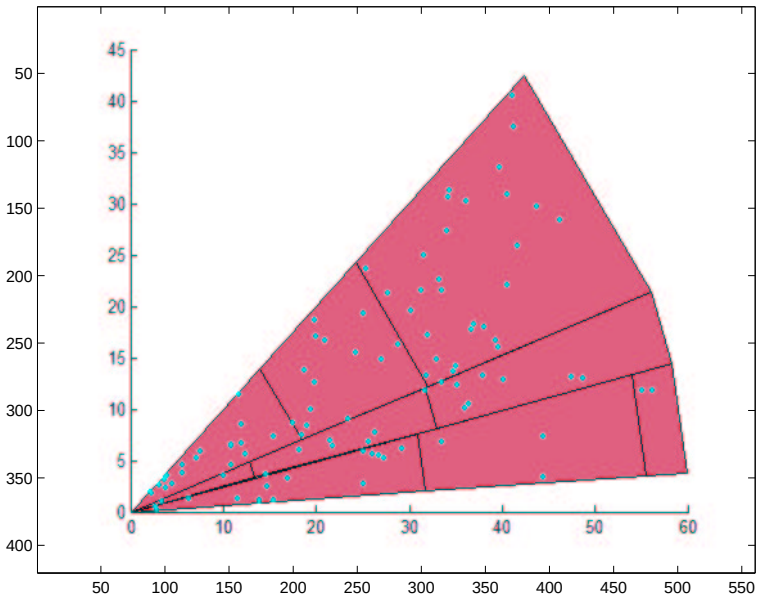

FIGURE 8.7 – Résultats pour le cas non uniforme : nr = 4 et nt = 4

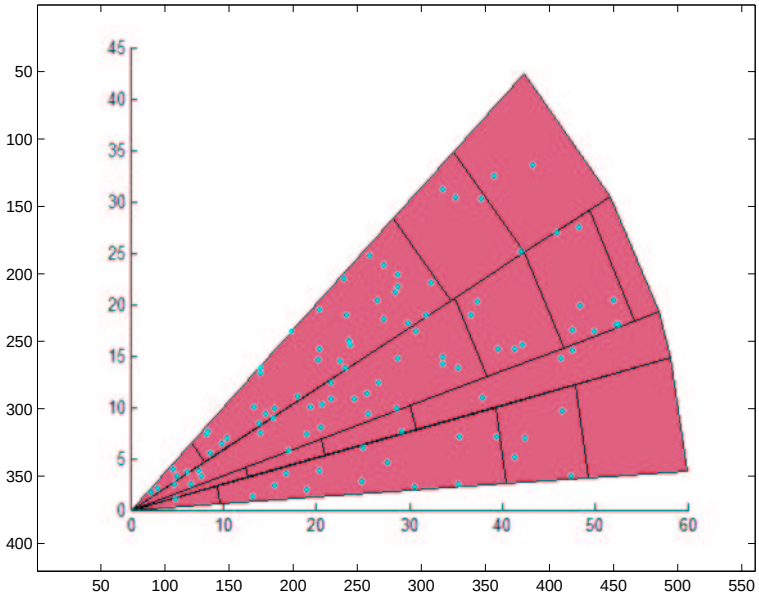

FIGURE 8.8 – Résultats pour le cas non uniforme : nr = 5 et nt = 5

Interprétation des résultats

L'interprétation globale revient à une visualisation shématique des résultats. Du fait que la région est subdivisée en districts où celles limitrophes au dépôt ou bien celles indexant plus de points de demande, se retrouvent moins éparses que les autres, tend à confirmer la crédibilité des résultats obtenus.
L'économie du carburant et le gain en heures de travail supplémentaire, en est une preuve indéniable.

8.3 Conclusion

Nous avons appliqués, au sein de ce chapitre, un algorithme hybride basé sur le couplage du gradient projeté perturbé avec l'algorithme génétique.
La complexité de la problématique logistique est un atout majeur d'expérimentation.
En ce sens, l'essentiel du problème de transport choisi relève de sa contenance en variables aléatoires comme le facteur humain par exemple.
L'implication de l'hybridation dans ce type de problèmes confère, de manière pragmatique, un solutionnement optimal.

Chapitre 9

Conclusion

Ce travail concerne un thème dont l'ensemble des facettes n'a pas été exploré à ce jour :
L'optimisation globale non convexe et non linéaire.
Nous retrouvons ce thème dans notre vie quotidienne tels, par exemple,

- Les problèmes d'économie et d'économétrie.
- Les problèmes à vocation agricole.
- Les problèmes mécaniques, en particulier le calcul de l'indice de fiabilité qui revient à minimiser une fonction objectif.
- Le design du système d'exploitation industrielle.

La résolution de problèmes d'optimisation globale non convexes et non linéaires reste, en général, un problème ouvert, pour lequel n'existe pas à cette date des méthodes aptes à résoudre l'ensemble des problèmes de ce domaine.
Nous avons abordé ce problème dans l'esprit de construire une procedure systématique sinon pour la résolution du moins pour la construction de méthodes numériques générales. Nous avons ainsi choisi l'approche *d'hybridation* qui permet de combiner aisément des approches de nature stochastique ou déterministe.
Dans ce cadre, nous avons envisagé les *perturbations stochastiques* des méthodes de descente. Nos efforts ont principalement porté sur les situations comprenant des restrictions, qui ont été abordées par :

- la méthode SQP ; méthode d'optimisation détérministe de type pénalité,
- la méthode de gradient projeté ; méthode d'optimisation détérministe de

type projection,

Pour certaines situations, les résultats fournis par ces méthodes ne sont pas satisfaisants ; tel est le cas, par exemple, des situations non convexes.

Dans ce cas, une approche purement stochastique peut, en théorie, déterminer l'optimum global. Ces méthodes sont très générales, car ne requièrent aucune hypothèse contraignante sur la fonction objectif ou sur les contraintes.

Cependant, ces méthodes ne convergent que très lentement et sont peu efficace numériquement. Leur couplage avec des procedures déterministes se revèle plus efficace : l'application de cette méthodologie à des problèmes classiques donne des résultats de meilleure qualité.

L'application à un problème de transport stochastique s'est également révélée probante et tendant à établir l'efficacité de l'approche par hybridation.

Ces essais ouvrent une voie pour la résolution des problèmes envisagés. Pour l'avenir, nous souhaiterions proposer cette approche aussi bien aux industriels qu'aux agriculteurs, que pour l'ensemble des composants du tissu économique, pour pouvoir émanciper leurs activités respectives.

Bibliographie

[1] M. Balachandran. Knowledge-Based Optimum Design, Topics in Engineering Vol. 10. Southampton : Computational Mechanics Publications, 1993. BU : 006.3 BAL

[2] P. Basso. Iterative Method for the Localization of the Global Maximum, SIAM Journal on Numerical Analysis 19, 781-792, 1982.

[3] P. Basso. Optimal Search for the Global Maximum of Function with Bounded Semi-norm, SIAM Journal on Numerical Analysis 22, 888-903, 1985.

[4] C. Bastian, A.K. Rinnooy Kan. The stochastic vehicle routing problem revisited. European Journal of Operational Research, 1992, vol 56, pp. 407-412.

[5] T. B*ä*ck and F. Hoffmeister. Global optimization by means of evolutionary algorithms. A.N. Antamoshkin, editor, Random Search as a Method for Adaptation and Optimization of Complex Systems, p. 17-21, Divnogorsk, ex-URSS, mars 1991.

[6] D. Beasley, D.R. Bull et R.R. Martin. An Overview of Genetic Algorithms : Part 1, Fundamentals. University Computing, vol. 15, n°2, p. 58-59, 1993a.

[7] D. Beasley, D.R. Bull et R.R. Martin. An Overview of Genetic Algorithms : Part 2, Research Topics. University Computing, vol. 15, n°4, p. 170-181, 1993b.

[8] M. Bouhadi. Contribution à l'Optimisation Globale avec Contraintes. Approche Stochastique. Thesis, Université Mohammed V, Faculté des Sciences, Rabat, Maroc, 1997.

[9] M. Bouhadi, J.E. Souza De Cursi, R. Ellaia. Global Optimization under nonlinear restrictions by using stochastic perturbations of the projected gradient. Floudas, Christodoulos A. ; Pardalos, P.M.. (Org.). Frontiers in Global Optimization. Dordrecht : Springer, 2003, v. 1, p. 541-561.

[10] N. Bouleau. Processus Stochastique et Applications. Hermann, Paris. 1988.

[11] Charles Darwin. The origin of Species. 1859.

[12] P. G. Ciarlet. Introduction à l'Analyse Numérique Matricielle et à l'Optimisation, 1990, Masson, Paris.

[13] C. Coello. Use of a Self-Adaptive Penalty Approach for Engineering Optimization Problems. Computers in Industry, 2000, 41(2) :113-127.

[14] Y. M. Danilin and S.A. Piyavskii. An Algorithm for Finding an Absolute Minimum, Theory of Optimal Solutions. IK Akad.Nank, USSR, Kiev, 25-37, 1967. (Russian)

[15] R. Ellaia. Contributions à l'Optimisation Globale et à l'Analyse Non Différentiable. These-ès-sciences, Université Mohammed V, Faculté des sciences, Rabat, Maroc, 1992.

[16] M. Z. Es-sadek, R. Ellaia et J. E. Souza De Cursi. Application of an hybrid algorithm in a logistic problem. Journal of advanced research in applied mathematics. Vol. 1, Issue. 1, 2009, pp. 34-52.

[17] Y. G. Evtushenko. Numerical Methods for Finding the Global Extremum of a Function. USSR Computational Mathematical Physics 11, 38-54, 1971.

[18] R. Fletcher. Practical Methods of Optimization. John Wiley and Sons, 1987

[19] E. A. Galperin. The Cubic Algorithm, Journal of Mathematical Analysis and Applications 112(2), 635-640, 1985.

[20] L. Goffe, G.D. Ferrier and J. Rogger. Global optimization of statistical functions with simulated annealing. Journal of Econometrics, Vol. 60, pp. 65-99.. 1994

[21] D. Goldberg. Genetic algorithms. Addison Wesley editor. 1989.

[22] D.E. Goldberg. Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley Publishing Company, INC, 1989.

[23] S.P. Han. A globally convergent method for nonlinear programming. J. Optimization Theory and Applications, vol. 22, pp. 297-309, July 1977.

[24] P. Hansen. Global Optimization Using Interval Analysis. Marcel Dekker, 1992.

[25] P. Hansen, B. Jaumard and S. H. Lu. Global Optimization of Univariate Lipschitz Functions : I.Survey and Properties, Mathematical programming 55, 251-272, 1992.

[26] P. Hansen, B. Jaumard and S. H. Lu. Global Optimization of Univariate Lipschitz Functions : II.New Algorithms and Computational Comparaison, Mathematical programming 55, 273-292, 1992.

[27] P. Hansen and B. Jaumard. Lipschitz Optimization, Les Cahiers du GERARD, May 1994.

[28] J. Holland. Adaption in Natural and Artificial Systems. 1975

[29] R. Horst et H. Tuy. Global Optimization-Deterministic Approaches, 2nd Edition, Berlin : Springer, 1992.

[30] R. Horst and H. Tuy. On the Convergence of Global Methods in Multi extremal Opti- mization, Journal of Optimization Theory and Applications 54, 253-271, 1987.

[31] P. Jaillet, A. Odoni. The probabilistic vehicle routing problème. B.L. Golden and A.A. Assad (eds.) Vehicle Routing : Methods and Studies, North-Holland, Amsterdam, 1988.

[32] S. Kirkpatrick, C.D. Gelatt and M.P. Vecchi. Optimization by Simulated Annealing. Science, May 1983.

[33] J.B. Lasserre, Prieto, Rumeau SDP Vs. LP Relaxations for the moment approach in some performance evaluation problems. Rapport LAAS N03125, Mars 2003, 16p.

[34] A.V. Levy, A. Montalvo. The tunneling algorithm for the global minimization of functions. SIAM J. Sci. Stat. Comput, 1985.

[35] D. Q Mayne and E. Polak. Outer Approximation Algorithm for Non differentiable Optimization Problems, Journal of Optimization Theory and Applications 42, 19-30, 1984.

[36] C. C. Meewella and D. Q. Mayne. An Algorithm for Global Optimization of Lipschitz Functions, Journal of Optimization Theory and Applications 57, 307-323, 1988.

[37] R. H. Mladineo. An algorithm for Finding the Global Maximum of Multimodal, Multi- variate Function, Mathematical Programming 34, 188-200, 1986.

[38] J.P. Nougier. Méthodes de calcul numérique. Paris : Masson, 1987. BU : 519.4

[39] A.G. Novaes, O.D Graciolli. Designing multi-vehicle delivery tours in a gridcell format. European Journal of Operational Research, 1998.

[40] A.G. Novaes, J.E. Souza De Cursi, O. Graciolli. A continuous approach to the design of physical distribution systems. Computer And Operations Research, 2000, v. 27, n. 9, p. 877-893.

[41] Ji.M. Peng, Y. Yuan. Optimality conditions for the minimization of a quadratic with two quadratic constraints, SIAM Journal of Optimization 7 (1997) 579594.

[42] J. Pinter. Global Optimization in Action, Kluwer, Dordrecht, Deutschland, 1996.

[43] S.A. Piyavskii. An algorithm for finding the absolute extremum of a function, USSR Computational Mathematics and Mathematical physics, 12, 1972, pp.57-67 (Zh. vychisl Mat.mat.Fiz. 12(4) (1972) 888-896)

[44] M. Pogu and J.E. Souza de Cursi. Global Optimization by Random Perturbation of the Gradient Method with a Fixed Parameter, 1994, Journal of Global Optimization Vol. 5, 159–180.

[45] K. Popper. Toute vie est résolution de problèmes. Actes Sud, 1997.

[46] M.J.D. Powell. Algorithms for nonlinear constraints that use Lagrangian functions. Math. Programming, vol. 14, pp. 224248, 1978.

[47] M.J.D. Powell. Variable Metric Methods for Constrained Optimization. Mathematical Programming : The State of the Art, (A. Bachem, M. Grotschel and B. Korte, eds.) Springer Verlag, pp 288-311, 1983.

[48] I. Rechenberg. Evolution strategies. (Evolutions strategie in original). 1960.

[49] M.J. Rijckaert, X.M. Martens. Comparison of generalized geometric programming algorithms. Journal of Optimization Theory and Application, 1978, 205-241.

[50] J.B. Rosen. The gradient projection method for nonlinear programming. Part II. Nonlinear constraints. J. Soc. Indust. Appl. Math, 1961, Vol 9 No 4, aa-bb.

[51] R. Sanguthevar. On Simulated Annealing and Nested Annealing, Dept. of Computer and Information Science and Engineering, University of Florida, Gainesville, FL 32611.

[52] S. Schaffler and K. Ritter. A Stochastic Method For Constrained Global Optimization. 894–904. SIAM J. Optimization. 1994.

[53] K. Schittkowski. NLQPL : A FORTRAN-Subroutine Solving Constrained Nonlinear Programming Problems, Annals of Operations Research, Vol. 5, pp 485-500, 1985.

[54] F. Schoen. On a Sequential Search Strategy in Global Optimization Problems, Calcolo 19, 321-334, 1982.

[55] Z. Shen and Y. Zhu. An Interval Version of Shubert's Iterative Method for the Localization of the Global Maximum, Computing 38, 275-280, 1986.

[56] B.O. Shubert. A Sequential Method Seeking the Global Maximum of a Function, SIAM Journal on Numerical Analysis 9, 379-388, 1972.

[57] Y. Smeers. Generalized Reduced Gradient Method as an Extension of Feasible Direction Methods, J. Optimization Theory Appl. 22, 209-226 (1977).

[58] J.E. Souza De Cursi, A. El Mouatasim, R. Ellaia. Random Perturbations of variable metric descent in unconstrained nonsmooth nonconvex perturbation. International Journal of Applied Mathematics and Computer Science, v. 16, p. 463-474, 2006.

[59] C. Wang et al. A new filled function method for unconstrained global optimization. Journal of Computational and Applied Mathematics 2008, doi :10.1016/j.cam.2008.07.001

[60] Q. Zheng, D. Zhuang. Integral global minimization : algorithms, implementations and numerical tests. Journal of Global Optimization, 1995, vol. 7, No. 4, 451-454.

[61] http ://w3.toulouse.inra.fr/centre/esr/CV/bontemps/WP/Algogene.html