Hybridations d'algorithmes métaheuristiques en optimisation globale et leurs applications

Hanaa Hachimi

Université Mohammed V - Agdal, Rabat
École Mohammadia d'Ingénieurs

Institut National des Sciences
Appliquées de Rouen

THÈSE DE DOCTORAT EN COTUTELLE

pour obtenir les grades de

DOCTEUR EN SCIENCES DE L'INGENIEUR
ECOLE MOHAMMADIA D'INGÉNIEURS,
UNIVERSITÉ MOHAMMED V AGDAL
Spécialité : Mathématiques appliquées et Informatique.
Option : Optimisation, Analyse numérique, Statistique.

et de

DOCTEUR DE L'INSTITUT NATIONAL DES SCIENCES APPLIQUÉES DE ROUEN
Spécialité : Optimisation des structures.
Option : Fiabilité, Qualité, Sécurité.

présentée par

Hanaâ HACHIMI

Sujet de la thèse :

HYBRIDATIONS D'ALGORITHMES MÉTAHEURISTIQUES EN OPTIMISATION GLOBALE ET LEURS APPLICATIONS

Soutenue publiquement le 29 **Juin** 2013 devant le jury composé de :

Pr. Nabil HMINA	Directeur, ENSA Kénitra	Président
Pr. Rajae ABOULAICH	Professeur, EMI Rabat	Rapporteur
Pr. Bouchaib RADI	Professeur, FST Settat	Rapporteur
Pr. Moussa KARAMA	Professeur, ENIT Tarbes	Rapporteur
Pr. Mohamed TKIOUAT	Professeur, EMI Rabat	Examinateur
Pr. Younes AOUES	Professeur, INSA Rouen	Examinateur
Pr. Rachid ELLAIA	Professeur, EMI Rabat	Directeur de thèse
Pr. Abdelkhalak EL HAMI	Professeur, INSA de Rouen	Directeur de thèse

Table des matières

Liste des tableaux

Table des figures

Remerciements

Nombreux sont ceux que je voudrais remercier pour m'avoir aidé, soutenu ou accompagné durant mes années de thèse. C'est pour leur montrer toute ma reconnaissance que je leur dédie ces quelques lignes.
Ce travail a été réalisé dans le cadre du CEDOC : Sciences et Techniques pour l'Ingénieur, au laboratoire LERMA de l'École Mohammadia d'Ingénieurs.
Je tiens à remercier tout particulièrement mon directeur de thèse Pr. Rachid ELLAIA, le premier qui m'a donné la chance d'entrer dans le monde de la recherche. Je voudrais lui témoigner toute ma reconnaissance pour sa disponibilité, sa rigueur et ses conseils.
Mon codirecteur Pr. Abdelkhalak EL HAMI, Professeur à L'INSA de Rouen, à qui je veux apporter mes remerciements tous particuliers. Il a su m'encadrer avec efficacité, clairvoyance et beaucoup de patience durant mes stages au sein du laboratoire LOFIMS ancien (LMR) de L'INSA de Rouen.
Je remercie également Pr. Nabil HMINA, Directeur de l'Ecole Nationale des Sciences Appliquées de Kénitra, de m'avoir fait l'honneur de présider le jury de ma soutenance. Je remercie Pr. Mohamed TKIOUAT, Professeur à l'EMI, et Pr. Younes AOUES, Professeur à l'INSA qui ont accepté d'examiner ce travail et de faire partie du jury.
Mes remerciements s'adressent également au Pr. Rajae ABOULAICH Professeur à l'EMI, Pr. Bouchaib RADI Professeur à la FST de Settat et Pr. Moussa KARAMA , Professeur à l'ENIT de Tarbes, pour avoir accepté d'évaluer mes travaux en qualité de rapporteurs.
Je tiens à remercier Mr.Mraoua, Mr.Zidani, Mme.Assif, Mr.Makhloufi, Mme.Iounes et Mr.Lemosse pour leur collaboration scientifique et amicale fructuese. Je voudrais profiter de ce manuscrit pour remercier tous les collègues du laboratoire LERMA, qui de près ou de loin ont contribué au bon déroulement de mes travaux tout le long des années de ma thèse. Mes remerciements s'adressent également aux membres du laboratoire LOFIMS (LMR) pour leur accueil et leur soutien.
A vous tous, je dis merci.

Résumé

L'optimisation des structures est un processus essentiel dans la conception des systèmes mécaniques et électroniques. Cette thèse s'intéresse à la résolution des problèmes mono-objectifs et multiobjectifs des structures mécaniques et mécatroniques. En effet, les industriels ne sont pas seulement préoccupés à améliorer les performances mécaniques des pièces qu'ils conçoivent, mais ils cherchent aussi à optimiser leurs poids, leurs tailles, ainsi que leurs coûts de production. Pour résoudre ce type de problème, nous avons fait appel à des métaheuristiques robustes, qui nous permettent de minimiser le coût de production de la structure mécanique et de maximiser le cycle de vie de la structure. Alors que des méthodes inappropriées de l'évolution sont plus difficiles à appliquer à des modèles mécaniques complexes en raison de temps calcul exponentiel. Il est connu que les algorithmes génétiques sont très efficaces pour les problèmes NP-difficiles, mais ils sont très lourds et trop gourmands quant au temps de calcul, d'où l'idée d'hybridation de notre algorithme génétique par l'algorithme d'optimisation par essaim de particules (PSO) qui est plus rapide par rapport à l'algorithme génétique (GA). Dans notre expérimentation, nous avons obtenu une amélioration de la fonction objectif et aussi une grande amélioration de la minimisation de temps de calcul. Cependant, notre hybridation est une idée originale, car elle est différente des travaux existants. Concernant l'avantage de l'hybridation, il s'agit généralement de trois méthodes : l'hybridation en série, l'hybridation en parallèle et l'hybridation par insertion. Nous avons opté pour l'hybridation par insertion par ce qu'elle est nouvelle et efficace. En effet, les algorithmes génétiques se composent de trois étapes principales : la sélection, le croisement et la mutation. Dans notre cas, nous remplaçons les opérateurs de mutation par l'optimisation par essaim de particules. Le but de cette hybridation est de réduire le temps de calcul ainsi que l'amélioration la solution optimale.

Mots clés : Optimisation, algorithmes métaheuristiques, algorithmes hybrides.

Abstract

This thesis focuses on solving single objective problems and multi-objective of mechanical and mechatronic structures. The optimization of structures is an essential process in the design of mechanical and electronic systems. Industry are not only concerned to improve the mechanical performance of the parts they design, but they also seek to optimize their weight, size and cost of production. In order to solve this problem we have used Meta heuristic algorithms robust, allowing us to minimize the cost of production of the mechanical structure and maximize the life cycle of the structure. While inappropriate methods of evolution are more difficult to apply to complex mechanical models because of exponential calculation time. It is known that genetic algorithms are very effective for NP-hard problems, but their disadvantage is the time consumption. As they are very heavy and too greedy in the sense of time, hence the idea of hybridization of our genetic algorithm optimization by particle swarm algorithm (PSO), which is faster compared to the genetic algorithm (GA). In our experience, it was noted that we have obtained an improvement of the objective function and also a great improvement for minimizing computation time. However, our hybridization is an original idea, because it is a different and new way of existing work, we explain the advantage of hybridization and are generally three methods : hybridization in series, parallel hybridization or hybridization by insertion. We opted for the insertion hybridization it is new and effective. Indeed, genetic algorithms are three main parts : the selection, crossover and mutation. In our case, we replace the operators of these mutations by particle swarm optimization. The purpose of this hybridization is to reduce the computation time and improve the optimum solution.

Keywords : optimization, meta heuristics algorithms, hybrid algorithms.

Introduction Générale

"Un bon croquis vaut mieux qu'un long discours" Napoléon 1er.

Cette thèse s'intéresse à la résolution des problèmes mono-objectifs et multiobjectifs des structures mécaniques et mécatroniques. Elle a été menée au sein de l'équipe **optimisation et analyse numérique** du **Laboratoire d'Étude et Recherche en Mathématiques Appliquées** (LERMA, École Mohammadia d'Ingénieurs, Université Mohammed V) en collaboration avec l'équipe d'accueil **optimisation fiabiliste des structures** du **Laboratoire d'Optimisation et FIabilité en Mécanique des Structures** (LOFIMS, Institut National des Sciences Appliquées de Rouen).

La conception optimale, appelée aussi optimisation de formes devient un outil indispensable pour presque tous les domaines industriels, notamment en mécanique de structures ; c'est une préoccupation essentielle des chercheurs qui conçoivent des objets industriels innovants (structures mécaniques, profils aérodynamiques, antennes, composants électroniques, etc.) afin d'en augmenter des propriétés physiques essentielles (solidité, efficacité, durabilité et fiabilité). Cette conception doit aussi satisfaire des contraintes strictes telles que le coût, la faisabilité industrielle, mais aussi le poids ou le volume des structures, ou bien toute autre propriété physique importante. Il s'agit donc d'un problème classique en mathématiques : l'optimisation sous contraintes d'une fonction, dite objectif. Grâce au développement de la puissance de calcul des ordinateurs, les mathématiques sont devenues essentielles dans l'automatisation de ce processus d'optimisation. En effet, la méthode traditionnelle d'optimisation est de procéder par essais et erreurs, suivant le savoir-faire et l'intuition du chercheur : on essaye une forme dont on calcule la performance puis, en fonction de cette dernière, on la modifie pour essayer de l'améliorer et on recommence jusqu'à obtention d'une forme satisfaisante (à défaut d'être optimale). Cette façon analytique de faire est très lente, coûteuse et imprécise. De plus en plus, elle est remplacée par des logiciels d'optimisation numérique qui représentent la forme par un nombre limité de paramètres descriptifs (généralement des points de contrôle sur les bords), et l'améliorent itérativement en faisant varier ces paramètres de manière automatique. C'est dans ces logiciels que

se cachent les mathématiques, parfois anciennes, mais le plus souvent extrêmement pointues et issues d'une recherche contemporaine. En particulier, dans le cadre de la mécanique du solide, des développements récents sur l'optimisation géométrique et topologique de formes ont eu un impact considérable dans l'industrie, notamment automobile et aéronautique.

Problématique

Dans le domaine de mécanique elle se présente une forte demande d'optimisation de formes ou de fiabilité des structures. Les problèmes étudiés dans le cadre de cette thèse se divisent en deux volets : problèmes monobjectifs et multiobjectifs. Pour résoudre ce type de problèmes, nous avons opté pour l'hybridation des métaheuristiques évolutionnaires, vue qu'elles sont efficaces et fiables.

Objectif de thèse

Plusieurs sont les domaines d'application des métaheuristiques qui sont généralement observés dans la nature soit par le comportement d'une population, essaim d'abeilles ou d'oiseaux, ou par un changement inattendu. En effet, ce travail consiste à optimiser des algorithmes stochastiques pour l'obtention des résultats plus efficace, et surtout les utiliser pour résoudre des problèmes en mécanique et en électroniques.

Contenu de la thèse

Le chapitre I dresse un état de l'art non exhaustif des méthodes d'optimisation mono-objectif, et multi-objectif en accordant une importance particulière aux méthodes utilisées dans nos travaux de recherche. Il aborde aussi les définitions générales des méthodes d'optimisation qui se divisent en deux volets déterministes et non déterministes. Nous donnons aussi brièvement les formulations des problèmes traités tout au long de ce manuscrit.

Le chapitre II présente d'une manière détaillée, les métaheuristiques les plus utilisées et nous mettons l'accent sur celles que nous avons étudiées dans le cadre de cette thèse : les algorithmes génétiques, la méthode de l'évolution différentielle et la méthode de l'optimisation par essaim particulaire.

Le chapitre III est consacré à la description des méthodes hybrides proposées pour résoudre des problèmes d'optimisation et aussi à donner la notion de l'hybridation en insertion. Il donne un panorama sur les différents types d'hybridation et il met l'accent surtout sur l'hybridation en

insertion que nous avons utilisée pour résoudre les problèmes mono objectifs et multi objectifs détaillés dans les chapitres suivants.

Le chapitre IV traite les problèmes mono objectifs que nous avons étudiés. Il est consacré à une présentation des problèmes mono objectifs dans les domaines de la mécatronique. Il s'agit de deux applications : Le premier volet du chapitre aborde le problème de l'optimisation de la position de la vis d'une carte électronique et le deuxième volet étudie le problème de l'optimisation du joint de brasure d'une carte électronique.

Dans le chapitre V, nous proposons l'étude des problèmes multi objectifs en appuyant sur l'utilisation des méthodes qui génèrent le front de Pareto par les deux méthodes NNCM et NBI (détaillées dans le chapitre 3), ainsi il traite deux problèmes mécaniques et mécatroniques, respectivement ; l'étude des pylônes haubanés et le problème de carte électronique.
Nous conclurons le manuscrit en synthétisant les principales contributions présentées, et en dénombrant de nouvelles perspectives sur la base des travaux effectués.
Cette thèse a donné lieu à 3 publications, 3 projets de publication et plusieurs communications dans des manifestations nationales et internationales.

Publications

1. HACHIMI H, ELLAIA R and EL HAMI A (2012). "A New Hybrid Genetic Algorithm and Particle Swarm Optimization". Key Engineering Materials, pp 115-125.
2. ASSIF S, HACHIMI H, AGOUZOUL M, ELLAIA R, EL HAMI A and AOUES Y (2013). "Optimization for an electronic card with a new hybrid method". Advanced Materials Research, pp 143-151.
3. HACHIMI H, ELLAIA R and EL HAMI A (2013). "A new hybrid genetic algorithm with particle swarm optimization and normal boundary intersection method for generating the Pareto frontier". Journal of Automation & Systems Engineering (accepted).

Travaux en cours

1. HACHIMI H, ASSIF S, AOUES Y, AGOUZOUL M, EL HAMI A and ELLAIA R (soumis 2013). Optimization by heuristic algorithm for the solder joints of an electronic card, Optimization and Engineering.
2. HACHIMI H, IOUNES A, ELLAIA R LEMOSSE D, SUZA E and EL HAMI A (2013). Optimisation multi-objective d'un pylône haubané.
3. HACHIMI H, MAKHLOUFI A, ELLAIA R and EL HAMI A (2013). Optimization of engineering structures by heuristic algorithms.

Chapitre 1

L'Optimisation :Etat de l'art

1.1 Introduction

La résolution de problèmes d'optimisation est devenue un sujet central en recherche opérationnelle, le nombre de problèmes d'aide à la décision pouvant être formalisés sous la forme d'un problème d'optimisation étant en forte croissance[53]. Les problèmes d'apprentissage de réseaux de neurones, de planification des tâches ou encore d'identification sont, par exemple, des problèmes communs d'optimisation.
En outre, l'Homme cherche à améliorer sa vie quotidienne, l'Homme aime la perfection ! Et sans qu'il se rend compte, il essaye à minimiser ses charges, son loyer ou la consommation de sa voiture....Il tente toujours à vrai dire à optimiser que se soit minimiser ses dépenses et maximiser ses biens.
Le mathématicien vient pour concrétiser le voeux de l'Homme en modélisant les problèmes de la vie sous des fonctions coûts en utilisant les différents types d'optimisation.
De nos jours l'optimisation est devenue un domaine indispensable pour résoudre plusieurs problèmes que se soit dans l'industrie ou d'autres secteurs [51].
En effet nous avons assisté ces dernières années à une croissance très rapide des travaux utilisant les méthodes d'optimisation. Cette tendance peut être observée dans tous les domaines de la science.
Nous aborderons dans ce chapitre les définitions générales des méthodes d'optimisation qui se divisent sous deux volets déterministes et non déterministes. Nous donnons aussi brièvement les formulations des problèmes traités tout au long de ce manuscrit.

1.2 Problèmes d'optimisation

La formulation des problèmes d'optimisation reste très ambigüe à cause de la diversité des vocabulaires et des confusions éventuelles que cela pourrait engendrer. Nous avons convenu d'adopter le vocabulaire qui suit :

Un **problème d'optimisation mono-objectif** est défini par un ensemble de variables, une fonction objectif et un ensemble de contraintes.

Un **problème d'optimisation multiobjectif** est défini par un ensemble de variables, un ensemble de fonctions objectif et un ensemble de contraintes.

L'**espace d'état**, appelé aussi domaine de recherche, est l'ensemble des domaines de définition des différentes variables du problème.

Les **variables** du problème dite aussi variable de **conception** ou de **décision** peuvent être de nature diverse (réelle, entière, booléenne. etc.) et exprimer des données qualitatives ou quantitatives, dans la présente thèse on s'intéresse au cas réel.

La **fonction objectif** ou encore (fonction de coût) définit le but à atteindre, on cherche à minimiser ou à maximiser celle-ci.
Une fonction **multimodale** présente plusieurs minima (locaux et globaux). Tandis qu'une fonction **unimodale** n'a qu'un minimum, le minimum global.
L'ensemble des **contraintes** est en général un ensemble d'égalités ou d'inégalités que les variables de l'espace d'état doivent satisfaire. Ces contraintes limitent l'espace de recherche.
Les méthodes d'optimisation recherchent un point ou un ensemble de points dans l'espace de recherche qui satisfont l'ensemble des contraintes, et qui maximisent ou minimisent la fonction objectif.

Problème d'optimisation mono-objectif
Mathématiquement, dans le cas d'une minimisation, un problème d'optimisation mono-objectif se présente sous la forme suivante :

$$PO \quad \begin{cases} Minimiser \;\; f(x) \\ \text{sous les contraintes} \\ h_i(x) = 0 \quad , i = 1, ..., m, \\ g_j(x) \leq 0 \quad , j = 1, ..., k, \\ x_L \leq x \leq x_U \end{cases} \tag{1.1}$$

avec $x \in \mathbb{R}^d$ est le vecteur des variables de décision, f la fonction objectif, $h_1, .., h_m$ et $g_1, .., g_k$ sont respectivement les contraintes d'égalités et d'inégalité et x_L, x_U sont respectivement les bornes inférieures et supérieures du domaine de recherche des variables. Cet ensemble définit l'es-

pace d'état, tandis que l'ensemble de points de l'espace des états possibles qui satisfait au mieux les contraintes est donné par :

$$C = \{x \in \mathbb{R}^d / h_i(x) = 0,\ g_j(x) \leq 0\ and\ x_L \leq x \leq x_U\}$$

Il est possible de passer d'un problème de maximisation à un problème de minimisation grâce à la propriété suivante :

$$\max_{x \in C} f(x) = \min_{x \in C} -f(x)$$

Suivant la nature de $f(x)$, des contraintes $g_j(x)$ et des variables x, le problème d'optimisation correspondant porte des noms divers. On distingue plus particulièrement les cas suivants :

La Programmation Linéaire

Comme le nom l'indique la fonction objectif et les contraintes sont toutes linéaires

$$\begin{cases} \min \quad f(x) = \sum_{i=1}^{d} a_i x_i \\ \text{s. t.} \quad g_j(x) = \sum_{i=1}^{d} c_{ji} x_i \quad , j = 1, ..., k. \end{cases} \tag{1.2}$$

La Programmation Quadratique

Dans ce cas, on a

$$\begin{cases} \min \quad f(x) = c + \sum_{i=1}^{d} a_i x_i + \sum_{i,j=1}^{d} a_{ij} x_i x_j \\ \text{s. t.} \quad g_j(x) = \sum_{i=1}^{d} c_{ji} x_i \quad , j = 1, ..., k \quad x_i \geq 0,\ \ i = 1, ..., n. \end{cases} \tag{1.3}$$

C'est-à-dire $f(x)$ quadratique et $g_j(x)$ linéaire.

La Programmation Convexe

La fonction objectif et les contraintes sont toutes convexes ($f(x)$ et $g_j(x)$ sont convexes).

La Programmation Discrète

où les variables prennent des valeurs entières.

Les Problèmes de la Programmation continue

où les variables prennent des valeurs réelles, on a $C \subset \mathbb{R}^d$.

La Programmation Non Linéaire

La fonction objectif et les limitations sont non linéaires. Pratiquement ils recouvrent toutes les catégories, ce qui fait apparaître les autres comme des cas particuliers.

Derrière toutes ces catégories, il y a bien entendu des méthodes mathématiques de résolution qui en général portent le même nom. Dans ce qui suit, nous nous limiterons aux méthodes de programmation non linéaire qui sont directement en relation avec l'optimisation des structures.

Optimum global, optimum local

Soit un problème d'optimisation mono-objectif PO et C l'ensemble des solutions admissibles du problème.

- si l'on peut prouver que $\forall x \in C$, $f(xg) < f(x)$, alors on dira que xg est l'optimum (minimum) global du problème PO ;

-s'il existe un ensemble $V \subset C$, contenant $xloc$, tel que $\forall x \in V$, $f(xloc) < f(x)$, avec $x \neq xloc$, alors on dira que $xloc$ est un optimum (minimum) local du problème PO.

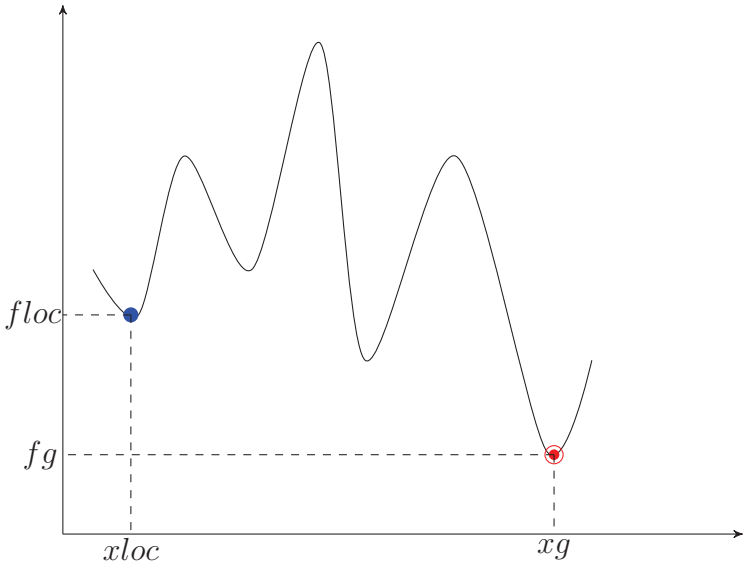

FIGURE 1.1 – Optimum global vs optimum local

1.3 Optimisation continue sans contraintes

1.3.1 Les méthodes d'optimisation locale

la recherche locale est une méthode utilisée pour résoudre des problèmes d'optimisation difficiles. La recherche locale peut être utilisée sur des problèmes de recherche d'une solution maximisant un critère parmi un ensemble de solutions candidates. Les algorithmes de recherche locale passent d'une solution à une autre dans l'espace des solutions candidates (l'espace de recherche) jusqu'à ce qu'une solution considérée comme optimale soit trouvée ou que le temps imparti soit dépassé. En revanche optimiser localement, c'est chercher une solution à un problème qui soit proche d'une solution de départ (optimisation locale), mais qui soit meilleure en terme de coût (fonction objectif). Pour cela, nous recherchons une meilleure solution par itérations successives, cette classe de méthodes peut être déterministe ou non-déterministe.

Optimisation déterministe

Lorsque l'évolution de la méthode de résolution est prévisible et ne laisse aucune place au hasard, celle-ci est qualifiée de **déterministe**.

Bien que l'optimisation locale ne soit pas le thème essentiel de notre travail. Nous en présentons les grandes lignes. En effet, comme nous le verrons plus loin beaucoup d'algorithmes d'optimisation globale utilisent des codes d'optimisation locale comme sous-programmes. Donc l'efficacité de ces derniers affecte celle des premiers.

1**- Rappels sur l'optimisation locale déterministe**

Cadre général

- **Position du problème**

Soit $f : \mathbb{R}^d \to \mathbb{R}$ une fonction continue, la formulation du problème d'optimisation locale s'écrit :

$$(Ploc) \quad \begin{cases} Trouver \quad (xloc, floc) \quad \in C \times \mathbb{R}, \quad où \quad floc = f(xloc), \\ \text{tel que :} \\ \forall \ x \in V, f(xloc) \leq f(x) \\ où \quad V \ est \ un \ voisinage \ de \ xloc. \end{cases} \tag{1.4}$$

Nous nous intéressons ici à l'aspect numérique de la résolution de $Ploc$. Pour ce faire, on va construire itérativement une suite $\{x_1, x_2, ..., x_n, ...\}$

de façon à trouver un point $xloc$ solution de $(Ploc)$.

La valeur $xloc$ qui minimise $f(x)$ peut ainsi être caractérisée de 2 manières :

i) $\forall x \neq xloc \qquad f(x) \geq f(xloc)$

ii) $\nabla f(xloc) = 0$ et $H(f(x))\backslash_{x=xloc}$ est une matrice définie positive [1]

La condition (i) a donné naissance à deux familles de méthodes dites d'**interpolation** et de **recherche directe** :

Les méthodes d'interpolation : elles sont utiles lorsque la fonction objectif est non définie en certains points de l'espace. Cette famille est composée de méthodes tentant de construire un modèle de la fonction à optimiser. Ce modèle est souvent un polynôme interpolant f, construit à partir d'un échantillonnage de points où f a déjà été évaluée. Le modèle est ensuite lui même optimisé. Les méthodes sans dérivées, ou sampling methods, elles sont décrites dans [[53],[54],[179],[180]]. Dans cette famille. Il existe des méthodes qui tentent d'approcher les gradients de f, et d'utiliser cette information afin de déterminer quels sont les points prometteurs où f est évaluée[57].

Les méthodes de recherche directe (direct search) : Le terme de recherche directe est utilisé pour décrire une méthode de recherche utilisant différents essais de solution possible en comparant chaque essai au meilleur résultat obtenu jusqu'à présent, combiné à une stratégie permettant de déterminer quelle est la prochaine solution à tester.

Cette famille de méthodes ne tente pas d'approcher le gradients ni de construire un modèle de f, des approximations de gradients sont utilisées pour ordonner les directions de recherche, elle se base principalement sur le calcul de la fonction objectif. Parmi ces méthodes se trouvent la méthode **simplexe** proposée par[60]. Des extensios de cet algorithme ont été proposées par **Nelder et Mead**[172], plusieurs variantes existent [[179],[180],[184],[183]] en ajoutant d'autres mouvements : en plus de la réflexion, ils proposent la contraction, la réduction et l'étirement. Ily a aussi l'algorithme de **Hooke et Jeeves** , l'algorithme **DIRECT** la **recherche par coordonnées**[64], **GPS** [[15], [31], [228]], [16],etc.

L'algorithme général des méthodes de recherche directe est le suivant :

1. $\varphi^T H \varphi > 0 \quad \forall \varphi \neq 0$.

Etape 1 : Initialisation
x_0
Etape 2 : Affectation
$x_{opt} \longleftarrow x_0$
Etape 3 : Itération
Tant que test d'arrêt n'est pas vérifié faire (1.5)
Mise à jour de x_k
Si $f(x_k) < f(x_{opt})$
$x_{opt} \longleftarrow x_k$
FinSi
Fin du Tant que

FIGURE 1.2 – Algorithme général des méthodes de recherche directe

En principe la mise à jour se fait selon une direction de recherche d_k par le processus itératif suivant

$$x_{k+1} = x_k + \lambda_k d_k.$$

avec λ_k est un pas de déplacement et x_{k+1} est l'approximation de $xloc$ à l'itération $k+1$.

Parmi les méthodes directes, ils existent des méthodes dont les algorithmes n'obéissent pas vraiment au schéma de l'algorithme général, c'est-à-dire dont les directions de recherche ne sont pas choisies de la façon définie ci-dessus. En effet, dans la **recherche de Fibonacci** l'algorithme commence par un intervalle contenant $xloc$, puis cet intervalle est réduit successivement en se basant seulement sur les évaluations de la fonction en ces bornes jusqu'à obtenir un intervalle suffisamment petit, ainsi, le minimum de la fonction peut être obtenu au centre de cette intervalle. Aussi, le "polytope mouvant" de **Nelder et Mead** où partant de $N+1$ points (pour le cas N- dimensionnel), nous effectuons des opérations géométriques sur ces points et nous remplaçons au fur et à mesure les points inutiles (points de très grande valeur), jusqu'à la convergence. Cette liste n'est pas exhaustive et elle n'a pas pour objectif de l'être mais plutôt de mettre en évidence les caractéristiques de chaque famille.

La condition (ii) donne naissance à une famille de méthodes dite de type **gradients** :

Les méthodes de type gradients :

Elles utilisent les gradients et/ou les Hessiens de f (ou leurs approximations) pour choisir des d_k et λ_k appropriés, de façon que la direction

choisie soit une direction de descente, elles sont appelé aussi **méthodes géométriques**.

direction de descent

Soit $f : \mathbb{R}^n \to \mathbb{R}$, une fonction continûment différentiable, et x un vecteur de $\mathbb{R}^n$. Le vecteur $d \in \mathbb{R}^n$ est appelé direction de descente de f en x ssi $\nabla f(x)^T d < 0$.

Le principe général est le suivant : x_1 donné, nous construisons un processus x_k satisfaisant

$$f(x_{k+1}) < f(x_k) \quad avec \quad x_{k+1} = x_k + \lambda_k d_k$$

où d_k est une direction de descente et λ_k minimise $f(x_{k+1})$ dans la direction d_k.

Finalement l'algorithme général qui sert à résoudre le problème Ploc peut s'écrire comme suit :

Initialisations
$x_0, \ \lambda_0, \ d_0$
Itération
Tant que test d'arrêt n'est pas vérifié faire
$$x_{k+1} = x_k + \lambda_k d_k$$
Fin du tant que

(1.6)

FIGURE 1.3 – Algorithme général des méthodes de descente.

Pour calculer d_k et λ_k, l'algorithme peut utiliser des informations sur les valeurs du gradient (voire du Hessien) de f au point x_k courant. Le choix de la direction de descente conduit à attacher à l'algorithme général des noms particuliers, tels qu'on le verra juste après.

2**- Quelques exemples d'algorithmes de descente**

Les méthodes de descente sont celles où les λ_k et d_k sont supposés être ceux qui donnent une direction de descente et un pas optimal. Comme il est signalé juste en haut le choix de la direction de descente a conduit au développement de plusieurs algorithmes, chacun avec ses propres caractéristiques :

• **L'algorithme du gradient à pas fixe** : C'est la méthode la plus simple, qui consiste à construire le processus itératif suivant

$$\left\| \begin{array}{cc} x_1 & donné \\ x_{k+1} = x_k - \lambda \nabla f(x_k) \end{array} \right. \tag{1.7}$$

où λ est appelé pas de descente, est une constante positive donnée, ne dépendant pas de k. La direction de descente est celle opposée à la direction du gradient.

La convergence de la méthode n'est assurée que pour de bonnes valeurs de λ, c'est-à-dire ni trop grandes ni trop petites. Quand cette convergence existe, elle est généralement lente, voire très lente.

Ces considérations de lenteur ne doivent cependant pas faire rejeter a *priori* la méthode car dans certaines situations elle est la seule à pouvoir être facilement et rapidement mise en oeuvre. En prenant de petites valeurs de λ, en effet la descente doit être vérifiée, c'est-à-dire que $f(x_{k+1}) < f(x_k)$.

• **L'algorithme du gradient à pas prédéterminé** : Il s'agit d'une amélioration de la méthode précédente. La direction de descente est toujours le gradient et λ_k est choisi tel que :

$$\lim_{k \longrightarrow +\infty} \lambda_k = 0$$

$$\sum_{k=0}^{+\infty} \lambda_k = +\infty$$

On démontre que sous ces hypothèses le processus converge.

• **L'algorithme du gradient à pas optimal (Steepest descent)** : Il est également connu sous le nom d'**algorithme de la plus forte pente** ou de **la plus profonde descente**. La méthode consiste à choisir le gradient comme direction de descente et à calculer, à chaque itération, la meilleure profondeur de descente. Donc l'algorithme est le suivant :

$$\left\| \begin{array}{cc} x_1 & donné \\ x_{k+1} = x_k - \lambda_k \nabla f(x_k) \end{array} \right. \tag{1.8}$$

où λ_k est calculé de manière à minimiser $f(x_{k+1})$ dans la direction du gradient. nous vérifions en effet qu'à l'étape k, la fonction $f(x_{k+1})$ ne dépend que de λ_k car $\nabla f(x_k)$ est fixé. Nous envisageons 2 cas

$\forall x \in \mathbb{R}^d$, nous pouvons observer le gradient $\nabla f(x)$

Nous utilisons le même processus itératif 1.8, où λ_k est une suite de nombres positifs vérifiant les conditions citées en haut.

$\forall x \in \mathbb{R}^d$, on ne peut observer $\nabla f(x)$ mais seulement $f(x)$ alors nous

utilisons une méthode de type **gradient approché**.

$$\left\| \begin{array}{ll} x_1 & donné \\ x_{k+1} = x_k - \lambda_k \nabla f(x_k) \end{array} \right. \tag{1.9}$$

Le gradient $\nabla f(x_k)$ peut être approché par la méthode de différence finie centrée. Pratiquement le plus souvent rencontré

$$\nabla f_i(x_k) = \frac{f(x_k + c_k e_i) - f(x_k - c_k e_i)}{2c_k} \quad pour \;\; i = 1, ..., d$$

($i^{ème}$ composante du vecteur $\nabla f(x_k)$ dans la base orthonormée e_i de $\mathbb{R}^d$)

• **L'algorithme de Newton** : Il s'agit d'une généralisation de la méthode de Newton-Raphson dans le cas mono-dimensionnel, c'est un algorithme efficace pour trouver des approximations d'un zéro (ou racine) d'une fonction d'une variable réelle à valeurs réelles. L'algorithme consiste à linéariser une fonction f en un point et à prendre le point d'annulation de cette linéarisation comme approximation du zéro recherché. Appliqué à la dérivée d'une fonction, cet algorithme permet d'obtenir une évaluation des points critiques.

Son algorithme est le même que celui de l'algorithme général avec un $\lambda_k d_k = -\dfrac{\nabla f(x_k)}{H(f(x_k))}$.

Cette méthode évite le calcul du pas optimal.

• **Les algorithmes quasi-Newtonien** :

L'idée est de remplacer H par une matrice judicieusement choisie. De nombreuses variantes existent, qu'on ne développe pas ici. Citons :
L'algorithme de Davidon-Fletcher-Powell,[[63],[88]].
L'algorithme de Broyden, Fletcher, Goldfarb, Shanno [[36], [37] [86],[107],[203],].

Remarque :

Ce qui différenciera les algorithmes basés sur de telles méthodes sera essentiellement :

- le choix de la direction : les gradients peuvent être utilisés, le hessien suivant le type de problème à résoudre.

- la façon de calculer le pas : souvent une autre minimisation unidi-

mensionnelle est utilisée tout le long de la direction de descente ; ce qui est alors appelé recherche linéaire.

- les critères d'acceptation des pas et directions pour éviter non seulement les divergences, mais aussi les lenteurs de convergence ou pour tenir compte d'éventuelles contraintes.

- et, bien sûr, le test d'arrêt de l'algorithme.

Optimisation stochastique

L'optimisation stochastique est une méthode qui génère et utilise des variables aléatoires. Ces variables aléatoires apparaissent dans la formulation du problème d'optimisation lui-même, qui impliquent des fonctions objectif aléatoires ou des contraintes aléatoires, par exemple. Certaines méthodes d'optimisation stochastiques aléatoires itérées sont utilisées pour résoudre des problèmes stochastiques, en combinant les deux significations de l'optimisation stochastique.
En effet les méthodes d'optimisation stochastique généralisent les méthodes déterministes pour les problèmes déterministes. De plus, une demande croissante existe pour la résolution de problèmes d'optimisation issus du monde réel, pour lesquels les données ou les fonctions objectif sont souvent incertaines, et où il s'avère parfois nécessaire de trouver des solutions robustes à des modifications ultérieures susceptibles de survenir sur les variables de décision. C'est la prise en compte de ces incertitudes dans le processus d'optimisation qui a donné naissance à l'optimisation stochastique.

Autrement dit, l'optimisation stochastique est un domaine de la programmation mathématique classique qui tient compte de l'incertitude du problème. La notion d'incertitude dans la programmation mathématique est apparue pour la première fois dans les années cinquantes. Elle a rencontré depuis un développement rapide.
Les avancées récentes ont ouvert de nouvelles perspectives qui devraient être fructueuses tant d'un point de vue théorique que pratique.

L'optimisation stochastique n'est pas une famille de problèmes, de modèles et d'outils différents des autres domaines d'optimisation (programmation linéaire, non-linéaire, dynamique) ; au contraire, elle constitue un complément de ces familles lorsque la notion d'incertitude intervient. Dans ce cas nous parlerons de programmation stochastique linéaire, non-linéaire ou bien de programmation stochastique dynamique. En effet, le but qui se fixe dans un problème d'optimisation stochastique est de prendre la meilleure décision vis à vis des situations qui comportent de l'incertitude.

La littérature autour de la programmation stochastique devient de plus en plus riche, les ouvrages les plus importants dans le domaine étant ceux de [129], [185] et [27].

1**- Introduction aux modèles de décision dans l'incertain**

Les approches qui dominent sur la modélisation des problèmes de la programmation stochastique sont les suivantes :

- Modèles avec des contraintes probabilistes,
- Modèles robustes.

a- Modèles avec des contraintes probabilistes

Ce sont des problèmes dont la modélisation appartient à la catégorie des modèles probabilistes, c.à.d. au lieu de considérer des contraintes déterministes nous travaillons plutôt avec des contraintes probabiliste, sans pour autant changer la fonction objectif qui reste, quand à elle, déterministe.

Le modèle mathématique s'écrit comme suit :

$$\begin{cases} \text{Minimiser } f(x) \\ \text{sous les contraintes} \\ h_i(x) = 0 \ , \quad i = 1, ..., m, \\ \mathbb{P}(\zeta \leq T\, x) \geq p \\ x_L \leq x \leq x_U \end{cases} \tag{1.10}$$

ζ est une variable aléatoire[2] et p est une probabilité réglée par l'utilisateur qui reflète une tolérance acceptable.
L'interprétation des contraintes $\mathbb{P}(\zeta \leq T\, x) \geq p$ reflète l'exigence que la contrainte $\zeta \leq T\, x$ soit satisfaite dans $p\%$ des cas, et pas nécessairement toujours. Surtout pour un problème où la non-satisfaction d'une contrainte est tolérée dans $p\%$ des cas, une exigence de $p = 100\%$ aurait été trop contraignante. Ceci est souvent le cas dans les problèmes de dimensionnement et de fiabilité où l'exigence par rapport aux probabilités de défaillance se modélise avec les contraintes probabilistes.

b- Modèles de Recours

L'idée principale ici est d'optimiser les décisions que l'on doit prendre au moment t_0 tout en tenant compte des conséquences de ces décisions pour toute réalisation des aléas qui pourrait se produire aux instants ultérieurs. Le mot **recours** révèle la possibilité dont nous disposons de remédier à une décision éventuellement trop optimiste à l'instant t_0, en mettant en oeuvre des actions correctives, évidemment plus chères, qui satisfont

2. Variable dont la valeur est déterminée en fonction du résultat d'une expérience aléatoire.

pourtant les contraintes posées aux instants ultérieurs $t_i,\ i > 0$.

On précise qu'après avoir pris la décision au moment t_0, nous passons au moment t_1 où tous les évènements inconnus jusqu'à cet instant se révèlent. Par conséquent, l'étude de tels cas a un intérêt seulement si nous disposons de quelques informations sur les réalisations des évènements qui nous sont inconnus au moment t_0. Toutes ces idées sont résumées dans le programme suivant :

$$\begin{cases} \text{Minimiser } \ c^T\ x + \mathbb{E}_\zeta(f(x,\zeta)) \\ \quad \text{sous les contraintes} \\ \quad\quad Ax = b \\ \quad \text{avec } \ \ f(x,\zeta) = \min\limits_{y \in \mathcal{Y}} q^T(\zeta) y(\zeta) \\ \quad \text{s. t. } \ \ T(\zeta)x + W(\zeta)y(\zeta) = h(\zeta) \end{cases} \tag{1.11}$$

où x et c sont des vecteurs de $\mathbb{R}^{n_1}$, A est une matrice de dimension $m_1 \times n_1$, $b \in \mathbb{R}^{m_1}$ le vecteur du second membre, et de façon similaire $y, q \in \mathbb{R}^{n_2}$, $h \in \mathbb{R}^{m_2}$ le vecteur aléatoire qui dépend de la variable aléatoire $\zeta \in \mathbb{R}^{r_1}$, $\mathbb{E}_\zeta$ est l'espérance mathématique par rapport à la variable aléatoire ζ, alors que T et W sont des matrices dépendant de la variable ζ et de dimensions appropriées ($T : m_2 \times n_1\ \ et\ \ W : m_2 \times n_2$).

Les variables x représentent les décisions avant que toute réalisation de la variable aléatoire ζ ne soit connue, alors que les variables y représentent les décisions prises, une fois que ζ sera observée. Nous cherchons à minimiser sur les variables de la première étape $x \in \chi$, mais nous veillons en même temps à ce que les coûts au second niveau, après la réalisation de la variable aléatoire ζ, soient minimaux en moyenne. Avec χ désigne un polyèdre convexe construit par un ensemble de contraintes linéaires.

c- Modèles robustes

Dans le cas où l'aléa n'intervient que dans la fonction objectif ; c'est l'optimisation robuste. Au fait, ce que l'on considère ici est la qualité de la solution obtenue et d'une certaine manière sa robustesse, à savoir les variations en termes de coûts pour différentes réalisations de la variable aléatoire. Cet aléa peut provenir de différentes sources (erreur de mesure sensorielle, simulations aléatoires). Pour ce genre de problème, à chaque appel de la fonction objectif avec les mêmes arguments, celle-ci retourne une valeur différente.

Le modèle mathématique s'écrit comme suit :

$$\begin{cases} \text{Minimiser } y(x,\zeta) \\ \text{sous les contraintes} \\ h_i(x) = 0 \quad , i = 1, ..., m, \\ g_j(x) \geq 0 \quad , j = 1, ..., k, \\ x_L \leq x \leq x_U \end{cases} \tag{1.12}$$

2**- Résolution numérique**

Approximations stochastiques

L'objet essentiel de la théorie de l'approximation stochastique est l'étude des algorithmes permettant de résoudre les modèles mathématiques précédents. L'optimisation stochastique, concerne l'optimisation d'une fonction f non directement observable.

On se propose de déterminer la valeur $xloc$ du paramètre x pour laquelle f est minimum. Nous distinguerons deux cas :

nous pouvons faire des observations bruitées de $\nabla f(x)$

On adopte le modèle :

$$\forall x \in \mathbb{R}^d \quad y(x,\zeta) = \nabla f(x) + \zeta(x)$$

Où y et ζ sont des Vecteurs aléatoires de $\mathbb{R}^d$.

$y(x)$ est directement observable pour tout x de $\mathbb{R}^d$ et $\zeta(x)$ est un bruit aléatoire vérifiant $\mathbb{E}(\zeta(x)) = 0, \forall x \in \mathbb{R}^d$.

Pour estimer la valeur $xloc$ du paramètre x pour laquelle $f(x)$ est minimale, le processus multidimensionnel d'approximation stochastique de **Robbins-Monro**est utilisé car il converge vers la solution de $\nabla f(x) = 0$.

nous pouvons faire des observations bruités de $f(x)$

C'est le cas le plus utilisé, alors le modèle suivant est adopté :

$$\forall x \in \mathbb{R}^d \quad y(x,\zeta) = f(x) + \zeta(x)$$

La détermination de $xloc$ se fera par une procédure du type **gradient approché stochastique**.

Méthode de gradient approché stochastique

On conserve le modèle $y(x) = f(x) + \zeta(x)$ où $\zeta(x)$ est un bruit aléatoire de moyenne nulle $\forall x \in \mathbb{R}^d$, $\mathbb{E}(\zeta(x)) = 0$. Pour estimer $xloc$, valeur du paramètre réel x, pour laquelle f est minimum, il suffit de trouver un

estimateur du $\nabla f(x)$. Dans ce même ordre d'idée, plusieurs chercheurs ont mis au point plusieurs processus intéressants dont deux d'entres eux ont attiré le plus d'attention :

a) Le processus de Kiefer-Wolfowitz
b) Le processus de James Spall

a) Le processus de Kiefer-Wolfowitz

Soit $\{e_1, ..., e_d\}$ une base orthonormée de $\mathbb{R}^d$ et soit $\{y(x), x \in \mathbb{R}^d\}$ une famille de variables aléatoires définies sur $\mathbb{R}^d$ et à valeur dans $\mathbb{R}$ tel que :

$\forall x \in \mathbb{R}^d \quad \mathbb{E}(y(x)) = f(x).$

Soit $\{\lambda_k\}$ et $\{c_k\}$ 2 suites de réels positifs. Le processus aléatoire multidimensionnel construit $\{X_k\}$ (X_k est un vecteur aléatoire de $\mathbb{R}^d$) :

nous définissons ainsi le processus de Kiefer-Wolfowitz, appelé souvent ($FDSA$, Finite Difference Stochastic Approximation) comme suit :

$$\left\| \begin{array}{ll} X_1 & \text{vecteur aléatoire donné} \\ X_{k+1} = X_k + \lambda_k W_k, & k \geq 1 \end{array} \right. \tag{1.13}$$

où W_k est un vecteur aléatoire, qui apparaît comme une approximation stochastique en différence finie centrée du gradient $\nabla f(x_k)$ et dont la $j^{ème}$ composante est définie par

$$W_k^j = \frac{y(x_k + c_k e_j) - y(x_k - c_k e_j)}{2c_k}, \;\; n \geq 1, \;\; j = 1, ..., d$$

La $FDSA$ est une méthode standard d'approximation stochastique, introduite en 1952 par Kiefer et Wolfowitz. Cependant, quand le nombre de dimension est grand la recherche devient de plus en plus laborieuse et l'algorithme converge en un temps énorme. En effet, si le problème est de dimension d alors en une seule itération pour évaluer le gradient il va falloir calculer la fonction objectif $2d$ fois.

Plus récemment, la méthode $SPSA$ a été introduite par James Spall en 1992, basée sur l'approximation stochastique du gradient par des perturbations simultanées. Cette méthode a connu tout récemment un fort regain d'intérêt, vu qu'elle nécessite moins de nombres d'évaluation de la fonction objectif comme il est défini dans ce qui suit.

b) Le processus de James Spall

Le processus de James Spall s'écrit comme suit :

$$\left\| \begin{array}{ll} X_1 & \text{vecteur aléatoire donné} \\ X_{k+1} = X_k + \lambda_k W_k, & k \geq 1 \end{array} \right. \tag{1.14}$$

où W_k est un vecteur aléatoire, qui peut être approché par l'estimateur suivant :

$$W_k^j = \frac{y(x_k + c_k\Delta_k) - y(x_k - c_k\Delta_k)}{2c_k\Delta_{kj}}, \;\; n \geq 1, \;\; j = 1, ..., d$$

avec Δ_k est un vecteur de d variables aléatoires indépendants symétriquement distribuées autour de zéro. En général nous travaillons avec la distribution symétrique de Bernoulli ± 1. Dans la littérature nous trouvons que des $\lambda_k = a/(A+k)^\alpha$ et $c_k = c/k^\gamma$, avec $a,\ c,\ A,\ \alpha$ et γ sont des coefficients non-négatifs fournissent de bon résultats.

Cette méthode ne requiert que deux évaluations de la fonction objectif par itération pour former le gradient et ceux indépendamment des dimensions du problème [209], [210]. .

Les limites des méthodes d'optimisation locale

Dans la catégorie de problèmes que nous avons présentés en haut, les méthodes cherchent le minimum d'une fonction en se basant sur la connaissance d'une direction de recherche. Bien sûr, ces méthodes seront toujours applicables et même recommandées lorsque la solution cherchée est réputée proche de la solution connue (point de départ) ou si la fonction objectif est convexe, en particulier, si le calcul n'est pas coûteux.

Toutefois, elles ne sont pas indiquées pour les problèmes multimodale où le risque de rester bloqué dans un optimum local est fort probable, cela suffit pour illustrer l'enjeu des problèmes d'autres types : L'optimisation globale. Quand un utilisateur se rend compte qu'il trouve des minimums locaux différents en exécutant son code d'optimisation (locale), il est tout à fait légitime qu'il songe à un code qui serait capable d'en tenir compte et de donner le meilleur de tous.
C'est ce genre de problèmes que **l'optimisation globale** propose de résoudre, ci après, un état de l'art des méthodes d'optimisation globale.

1.3.2 Les méthodes d'optimisation globale

Durant ces dernières années, l'optimisation globale a fait l'objet de plusieurs études grâce à des résultats théoriques nouveaux, une forte demande dans plusieurs domaines incluant des applications industrielles, et

au développement des moyens de calcul.

C'est ainsi que plusieurs articles ont été publiés sur le sujet traduisant la richesse des approches et des motivations ; nous citons à titre d'exemple[241], [230], [148], [158], [102],...etc. Contrairement à ce que l'on pourrait être tenté de croire, l'optimisation globale numérique n'a pas hérité (du moins pas systématiquement [3]) des facilités des techniques numériques d'optimisation locale. En effet, ces dernières utilisent pour la plupart des directions de descentes (conditionnées par des calculs de gradients ou de leurs approximations) comme nous avans vu, ce qui permet de converger naturellement vers un point de minimum local. Or, justement, l'optimisation globale évite de rester en de tels points. Il lui faudrait, au contraire, en échapper.

C'est pourquoi beaucoup d'approches ont été utilisées dans la tentative de résoudre le problème. Il s'agit, de trouver un état de minimum et de ne s'arrêter que s'il est le meilleur (l'optimum global). Pour illustration de cette diversité, nous citons dans la suite les méthodes les plus utilisées d'entre elles ainsi que les différents principes sur lesquels elles se basent.

En dépit de l'abondance des méthodes proposées, nous pouvons leur trouver des traits caractéristiques suivant leurs approches. Ainsi nous donnons la classification suivante : On distingue deux types d'approches selon qu'elles incluent ou non des processus probabilistes : **Les méthodes déterministes** et **les méthodes non-déterministes**.

Les méthodes déterministes

Ces méthodes ont tout d'abord été introduites pour résoudre de manière exacte des problèmes particuliers comme par exemple les problèmes continus et linéaires sous contraintes linéaires (algorithme du simplexe de Dantzig) ; ces méthodes ont aussi été élargies aux cas discrets et mixtes mais uniquement dans le cas linéaire.

La principale qualité des méthodes globales déterministes est qu'elles ne nécessitent pas de point de départ. Ces méthodes permettent de bien gérer les contraintes, contrairement aux méthodes stochastiques et peuvent s'appliquer aux problèmes mixtes (variables réelles, entières et de catégories). Elles garantissent l'obtention de la solution globale du problème. Cependant, il faut savoir que les méthodes déterministes globales restent utilisables tant que le nombre de variables ne devient pas trop important. Au delà d'une vingtaine de variables, elles atteignent leurs limites.

Ces méthodes permettent d'obtenir, à la convergence, la solution exacte du problème d'optimisation considéré avec une garantie absolue : en utilisant l'arithmétique d'intervalles arrondie telle qu'elle a été définie par [170]. Aucune erreur numérique insidieuse ne peut écarter de tels algorithmes de la solution optimale, il sera dans le pire des cas seulement ra-

3. méthodiquement.

lenti. Ces algorithmes sont appelés : algorithmes de Branch-and-Bound par intervalles . Les plus intéressantes d'entre elles sont :

• Méthodes à Recherche par Quadrillage (Grid Search Methods) ;
• Méthodes des Trajectoires (Trajectory Methods) ;
• Méthodes de séparation-évaluation (Branch-and-Bound).

Les méthodes Evolutionnaires et les méthodes non déterministes

Ces méthodes non-déterministes font appel à des tirages de nombres aléatoires. Elles permettent d'explorer l'espace de recherche plus efficacement. Dans la suite on s'intéressera plus particulièrement aux métaheuristiques.

Le mot métaheuristique est dérivé de la composition de deux mots grecs :

- heuristique qui vient du verbe heuriskein (euriskein) et qui signifie *trouver* ;

- méta qui est un suffixe signifiant *au-delà*, *dans un niveau supérieur*.

Les premières métaheuristiques datent des années 1980. Elles sont utilisées généralement quand les méthodes classiques échouent. Le terme métaheuristique est utilisé par opposition aux heuristiques. En effet, les métaheuristiques peuvent être utilisées pour plusieurs types de problèmes, tandis qu'une heuristique est adaptée à un problème donné. Les métaheuristiques ont comme caractéristiques communes de part leurs caractères stochastiques, c.à.d. qu'une partie de la recherche est conduite de façon aléatoire, elles sont inspirées d'analogies avec la réalité : physique (recuit simulé,...), biologie (algorithmes évolutionnaires, recherche tabou,...) ou éthologie (colonies de fourmis,...). En plus de cette base stochastique, les métaheuristiques sont généralement itératives, c-à-d qu'un même schéma de recherche est appliqué plusieurs fois au cours de l'optimisation, et directes, c-à-d qu'elles n'utilisent pas l'information du gradient de la fonction objectif. Elles tirent en particulier leur intérêt de leur capacité à éviter les optima locaux, soit en acceptant une dégradation de la fonction objectif au cours de leur progression, soit en utilisant une population de points comme méthode de recherche. Du fait du foisonnement de la recherche dans ce domaine, un grand nombre de méthodes de ce type existent. Dans la suite, nous verrons comment plusieurs concepts théoriques, notamment d'inspiration biologique, peuvent aider à la conception de nouvelles métaheuristiques.

- **Les méthodes inspirées des principes physiques**

Le recuit simulé : l'origine de cette méthode vient de l'analogie avec la métallurgie, elle consiste à monter la température du solide à des valeurs élevées pour atteindre des états de basse énergie d'un solide. Lorsque le solide est à une forte température, chaque particule possède une très grande énergie et peut effectuer de grands déplacements aléatoires dans la matière. Au fur et à mesure que la température est abaissée, chaque particule perd de l'énergie et sa capacité de déplacement se réduit. Les différents états transitoires de refroidissement permettent d'obtenir des matériaux très homogènes et de bonne qualité. Ce processus est appelé **recuit**.

Le recuit simulé a été établi indépendamment par [140], et [41] . L'idée de base est la suivante : à des paliers de températures décroissantes, l'algorithme utilise la procédure itérative , pour atteindre un état de quasi-équilibre thermodynamique. Cette procédure permet de sortir des minima locaux avec une probabilité d'autant plus grande que la température est élevée. Quand l'algorithme atteint les très basses températures, les états les plus probables constituent en principe d'excellentes solutions au problème d'optimisation.

Initialisation
Affectation
Tant que **le critère d'arrêt de la température** n'est pas satisfait faire
  Tant que (Repetition < **Palier_de_température**) faire
    Choisir s_n dans le **Voisinage** de s
    Calculer $DF \longleftarrow F(s_n) - F(s)$
    Si $DF < 0$ alors
      $s \longleftarrow s_n$
      Si $F(s_n) < F(s_{opt})$ alors
        $s_{opt} \longleftarrow s_n$
        $F_{opt} \longleftarrow F(s_n)$
      Fin Si
    Sinon
      $s \longleftarrow s_n$ avec une probabilité de $\mathbf{P}(\mathbf{T}, \mathbf{s}, \mathbf{s_{new}})$
    Fin Si
  fin du tant que
Mis à jour de la température T
Fin du tant que

(1.15)

FIGURE 1.4 – Algorithme général de recuit simulé

Au début du processus on **initialise** l'algorithme à une température élevée puis des mouvements aléatoires (**voisinage**) sont réalisés pour chaque

paramètre d'optimisation. Pour chacun de ces mouvements, la fonction objectif est évaluée et comparée avec la valeur antérieure. S'il y a un gain, les valeurs courantes des paramètres sont acceptées. Sinon, ils peuvent l'être quand même, avec une certaine probabilité dite de Bolztmann $\mathbf{P}(\mathbf{T}, \mathbf{s}, \mathbf{s_{new}}) = \mathbf{exp}(-\triangle\mathbf{E}/\mathbf{k_B}\ \mathbf{T})$, où $\triangle E$ est la différence entre les valeurs courantes et précédentes de la fonction objectif et T est le paramètre qui simule la température. Pour la suite on utilisera la loi classique c.à.d $T_k = \alpha T_{k-1}$, tel que α est appelé coefficient de refroidissement. Ce processus est répété pendant un nombre fixé $\mathbf{Palier\_de\_temprature}$, puis la température est réactualisée. La séquence des points générés converge normalement vers le minimum global.

En général, on choisit une température initiale suffisamment élevée qui donne une plus grande liberté pour l'exploration de l'espace de recherche. Puis, petit à petit, la température décroît jusqu'à atteindre une valeur proche de 0, ce qui signifie que la méthode n'acceptera plus de détériorer la solution.

• **Les méthodes inspirées par des comportements biologiques**

Les méthodes évolutionnaires ont un point commun puisque leurs itérations tendent à l'amélioration d'une solution unique. En effet, à partir d'une première solution, les approches décrites tentent de l'améliorer en fonction des contraintes du problème. Les méthodes auxquelles nous nous intéressons désormais considèrent la solution comme étant une population formée de plusieurs individus capables de répondre au problème.

a) L'algorithme de colonie de fourmis

Le premier algorithme d'optimisation par colonies de fourmis (*Ant colony optimization, ACO*) a été propose par Dorigo [73], [74]. vers le début des années quatre vingt dix ; . aussi appelé *Ant System (AS)*, a été développé spécialement pour résoudre le problème du voyageur de commerce. Depuis, cette approche a connu des variantes importantes et le nombre de travaux publiés augmente d'une année à l'autre, toute la théorie et les plus intéressants travaux récents sont dans [75, 227] et [204].

Le principe de la méthode provient analogiquement avec les comportements collectifs des insectes, les algorithmes de colonies de fourmis sont nés d'une constatation simple : les insectes sociaux, et en particulier les fourmis, résolvent naturellement des problèmes complexes. Un tel comportement est possible car les fourmis communiquent entre elles de manière indirecte par le dépôt de substances chimiques, appelées phéromones, sur le sol et le suivi de pistes observés dans les colonies de fourmis et construisent ainsi une solution à un problème en tenant compte de leur expérience collective. Au fait, elles adoptent pour la recherche de la solution la notion du **plus court chemin**.

D'une manière simplifiée, les fourmis commencent par se déplacer au

hasard. Puis, lorsqu'elles trouvent de la nourriture, elles retournent vers leur colonie, en marquant leur chemin à l'aide de phéromone. Si d'autres fourmis rencontrent ce chemin, il y a de fortes chances qu'elles arrêtent leurs déplacements aléatoires et qu'elles rejoignent le chemin marqué, en renforçant le marquage à leur retour, s'il mène bien vers la nourriture. Par conséquent, le chemin le plus court sera davantage parcouru, et donc plus renforcé et plus attractif. Par conséquent, le nombre de fourmis suivant cette trajectoire augmente. Au fil du temps, la quantité de phéromones déposée sur le plus long chemin diminue et finit par disparaître. Toutes les fourmis suivent alors le chemin le plus court.

L'algorithme de colonies de fourmis a été principalement utilisé pour produire des solutions quasi-optimales au problème du voyageur de commerce, puis, plus généralement, aux problèmes d'optimisation combinatoire. Récemment, son emploi se généralise à plusieurs domaines, depuis l'optimisation continue jusqu'à la classification, ou encore le traitement d'image.

Étape 1 **: Initialisation**
Initialiser les pistes de phéromone

Étape 2 **: Construction de la solution**
Pour chaque fourmi répéter
Construction de la solution en utilisant les pistes de phéromone

Étape 3 **: Mise à jour des pistes de phéromone**
Jusqu'à atteindre la condition d'arrêt

(1.16)

FIGURE 1.5 – Algorithme général de colonie de fourmis

b) Les algorithmes génétiques

Le concept d'**algorithme génétique** (de façon plus générale : algorithmes d'évolution) a été proposé par Holland en 1975 pour décrie les systèmes adaptatifs.

Les algorithmes génétiques, également appelés **algorithmes évolutionnaires**, sont inspirés du concept de sélection naturelle proposée par Charles Darwin. Le vocabulaire employé est directement calqué sur celui de la théorie de l'évolution et de la génétique. Nous parlerons donc ici d'individus (solutions potentielles), de populations, de gènes (variables), de chromosomes, de parents, de descendants, de reproductions, etc. Les points de l'espace de recherche sont alors les individus d'une population et la fonction à optimiser correspond à leur adaptation. Ces algorithmes font évoluer une population de manière itérative. Certains individus se repro-

duisent, d'autres mutent ou encore disparaissent et seuls les individus les mieux adaptés sont supposés survivre. L'héritage génétique des générations doit permettre à la population d'être de mieux en mieux adapté et donc de mieux répondre au critère d'optimisation.

Les principales étapes d'un algorithme génétique sont :

Étape 1 **: Génération de la population initiale**

Étape 2 **: Constitution d'une nouvelle population**
- Mesure de l'adaptation de chacun des individus
- Reproduction des individus en fonction de leur adaptation. Les plus performantes se reproduisent en priorité.
- Croisement de paires de séquences choisies aléatoirement
- Mutation de séquences tirées de manière aléatoire.

Étape 3 **: Critère d'arrêt**
- Il peut être défini en fonction du nombre de génération, de l'adapation du meilleur individu, etc.

(1.17)

FIGURE 1.6 – Les différentes étapes des algorithmes génétiques

En vue de sa généralisation et de son utilité dans plusieurs domaines industriels, maintes variantes de ces algorithmes ont vu le jour [69, 102, 230, 241].

c) La méthode des essaims particulaires

L'optimisation par essaim particulaire (OEP) est une métaheuristique d'optimisation née en 1995 aux États-Unis sous le nom de Particle Swarm Optimization (PSO). Elle a été inventée par Russel Eberhart (ingénieur en électricité) et James Kennedy (socio-psychologue). Initialement, ces deux concepteurs, cherchaient à modéliser des interactions sociales entre des « agents » devant atteindre un objectif donné dans un espace de recherche commun, chaque agent ayant une certaine capacité de mémorisation et de traitement de l'information. La règle de base était qu'il ne devait y avoir aucun chef d'orchestre, ni même aucune connaissance par les agents de l'ensemble des informations, seulement des connaissances locales. Un modèle simple fut alors élaboré.

Dès les premières simulations, le comportement collectif de ces agents évoquait celui d'un essaim d'êtres vivants convergeant parfois en plusieurs sous-essaims vers des sites intéressants. Ce comportement se retrouve dans bien d'autres modèles, explicitement inspirés des systèmes

naturels. Ici, la métaphore la plus pertinente est probablement celle de l'essaim d'abeilles, particulièrement du fait qu'une abeille ayant trouvé un site prometteur sait en informer certaines de ses consoeurs et que celles-ci vont tenir compte de cette information pour leur prochain déplacement Finalement, le modèle s'est révélé être trop simple pour vraiment simuler un comportement social, mais par contre très efficace en tant qu'outil d'optimisation.

Les différentes étapes de l'algorithme sont présentées dans la figure ci dessous :

$1-$ Initialiser la population de particules avec des positions et vitesses aléatoires.
$2-$ Evaluer la fonction objectif pour chaque particule et calculer g.
$3-$ Pour chaque individu i, L_i est initialisée à P_i.
$4-$ Répéter jusqu'au critère d'arrêt
    $4.1-$ Mettre à jour les vitesses et les positions des particules;
    $4.2-$ Evaluer la fonction objectif pour chaque individu;
    $4.3-$ Calculer les nouveaux L_i et g.
$5-$ Afficher le meilleur état rencontré au cours de la recherche.
(1.18)

FIGURE 1.7 – Les différentes étapes de la méthode des essaims particulaires

L'algorithme commence avec une initialisation aléatoire de l'essaim de particules dans l'espace de recherche. Chaque particule est modélisée par sa position dans l'espace de recherche et par sa vitesse. A chaque instant, toutes les particules ajustent leurs positions et vitesses, donc leurs trajectoires, par rapport à leurs meilleures positions, à la particule ayant la meilleure position dans l'essaim et à leur position actuelle. En réalité, chaque particule est influencée, non seulement par sa propre expérience, mais aussi par celle des autres particules. La position et la vitesse d'une particule dans un espace de recherche à N dimensions sont définies par : $P_i = (p_{i,1}, ..., p_{i,N})$ et $V_i = (v_{i,1}, ..., v_{i,N})$, respectivement. Chaque particule est caractérisée par sa meilleure position $L_i = (l_{i,1}, ..., l_{i,N})$ à l'itération t. La meilleure position qu'atteint l'essaim est sauvegardée dans le vecteur $G = (g_1, ..., g_N)$. La vitesse de chaque particule est mise à jour selon l'expression suivante :

$$v_{ij}(t+1) = K[w.v_{ij}(t) + c_1.r_1.(l_{ij} - v_{ij}(t)) + c_2.r_2.(g_j - v_{ij}(t))]$$

et

$$K = \frac{2}{|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}|}$$

où $\varphi = c_1 + c_2$ et $\varphi > 4$. Dans [47], les auteurs suggèrent les valeurs

suivantes $j = 1, ..., N$, w est une constante appelée facteur d'inertie, c_1 et c_2 sont des constantes appelées coefficients d'accélération, r_1 et r_2 sont des nombres aléatoires uniformément distribués dans l'intervalle $[0 \ \ 1]$. Si la vitesse calculée fait sortir une particule de l'espace de recherche, sa fitness n'est pas calculée. Compte tenu de la nouvelle vitesse, obtenue à partir de (a) et (b), la position à l'itération $t + 1$ est alors calculée :

$$p_{ij}(t + 1) = p_{ij}(t) + v_{ij}(t + 1), \qquad j = 1, ..., N.$$

Les limites des méthodes d'optimisation globale

Les algorithmes d'optimisation globale sont particulièrement efficaces lorsque la fonction est multimodale. Leur gros désavantage est que leur coût augmente, dans le pire des cas, exponentiellement avec la complexité du problème. Bellman a décrit ce concept en tant que malédiction dimensionnelle *curse of dimensionality*.

Bellman nous indique que si l'on veut trouver le minimum global d'une fonction de n_α variables avec une erreur relative sur les variables inférieure à ξ, alors il faudra effectuer $n_{eval} = \mathcal{O}((1/\xi)^{n_\alpha})$ évaluations. Par exemple, sur un problème de dimension $n_\alpha = 10$, si l'on veut localiser à 10% près les variables optimales, il faut $n_{eval} = (1/0.1)^{10}$ évaluations. Même pour une fonction évaluée en seulement 0.1 seconde, il faut plus de 10000 jours pour obtenir la solution. Afin de palier cet inconvénient, on assiste à une tendance vers les algorithmes hybrides.

1.3.3 Les méthodes d'optimisation hybride

L'utilisation des méthodes hybrides permet de combiner les avantages des deux types de méthodes (déterministes et non déterministes), elles peuvent être vu comme la solution parfaite vis-à-vis des désavantages et des méthodes locales et des méthodes globales. Au fait, les méthodes locales mènent à une solution locale. Tandis que les méthodes globales consomment énormément de temps, il parait claire qu'un compromis entre exploitation et exploration doit être trouvé d'où l'utilité des méthodes hybrides. Quand l'hybridation est basée sur une vraie maîtrise de l'idée derrière chacune des méthodes candidates, l'augmentation de la précision ainsi que la diminution du temps de calcul est assuré. Dans cet esprit, plusieurs types d'hybridation sont possibles. La méthode hybride des algorithmes génétiques avec la méthode des essaims particulaires de Kao [130] , ont préféré d'hybrider la méthode des essaims particulaires avec l'algorithme de colonie de fourmis selon des stratégies bien pensées par Shelokar [204]. Deux stratégies peuvent être utilisées pour expliquer le

principe d'hybridation :

Stratégie 1- On pourra par exemple s'approcher de l'optimum global par une méthode aléatoire puis affiner le résultat en appliquant successivement une méthode locale. Cette hybridation s'appelle l'hybridation bête parce qu'elle applique une méthode après l'autre. Il est claire que le résultat sera meilleur mais au détriment du temps de calcul, on consommera davantage de temps que si on avait appliquait les deux méthodes séparément. Un vrai sens d'hybridation, dans ce cas de figure, peut être d'appliquer une méthode globale et de trouver le bon moment de faire la commutation avec une méthode locale.

Stratégie 2- On pourra aussi essayer de trouver un état de minimum local et de n'y rester que s'il est le meilleur de minimus locaux (global). Cette stratégie a été utilisée dans trois méthodes d'optimisation globales suivantes :
La méthode des ***initialisations multiples*** ou (multistart). On utilise plusieurs fois une technique d'optimisation locale en différents points de C ; on prend le meilleur résultat ; il serait solution de PO.

La méthode du ***tunnelier*** (tunneling method). On cherche par une technique locale un minimum local x_l ; on utilise une technique (du Tunnelier) pour éliminer tous les points qui ont une valeur supérieure à celle de x_l (en effet ils ne pourraient être optimum global). Puis on réoptimise pour trouver un x_{l+1} et ... ainsi de suite. La suite des x_k ainsi construite converge vers le minimum global. Voir la figure ci dessous :

La méthode du ***montagnes russes*** (Steepest Ascent Descent : SAD.) D'abord on cherche un minimum local, puis on cherche cette fois-ci un maximum local ; cela permet d'échapper de l'état de minimum local. Puis on refait une minimisation suivie d'une autre maximisation et... ainsi de suite. Le plus petit minimum trouvé à la fin, est candidat pour être le minimun global.

1.3.4 Choix d'une méthode d'optimisation

On peut se demander à quoi sert une classification des types de problèmes et de méthodes. A notre avis quand un utilisateur est confronté à un problème d'optimisation globale, la première chose à faire est de bien cerner le problème, à savoir :

- Hypothèses sur f (différentiabilité, convexité...),
- hypothèses sur le domaine de recherche,
- existence ou non de contraintes, quels types de contraintes,

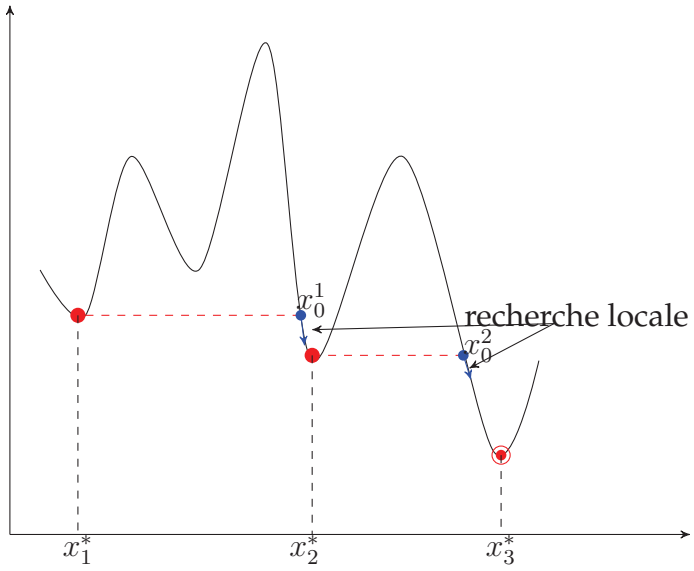

FIGURE 1.8 – La méthode de tunnelier

- coût d'évaluation de la fonction (temps CPU, nombre de sous programmes nécessaire),
- facilité de l'évaluation (accès, formule explicite de f),
- précision dont on dispose sur les calculs,
- type de matériel utilisé,
- temps dont on dispose pour résoudre le problème.

Donc, on voit bien, que connaître une classification des problèmes et des méthodes, peut faciliter la tâche de l'utilisateur et le guider dans son choix ; ce qui lui permet de fixer ses objectifs en conséquence. Car **il est préférable d'avoir une solution approchée (avec une précision insuffisante) dans un délai raisonnable qu'une solution exacte (avec la précision souhaitée) hors délais dans certains cas contractuels**.

Malheureusement, aucune méthode d'optimisation n'est capable de traiter efficacement tous les cas. En effet, [237]. ont démontré que si l'on considère l'ensemble de tous les problèmes d'optimisation possibles, alors aucun algorithme n'est meilleur qu'un autre (« no free lunch theorems for optimization »). Le choix d'un algorithme ne peut donc pas se faire par comparaison sur des cas tests généraux, mais doit aussi étudier des problèmes représentatifs des applications envisagées. La comparaison d'algorithmes d'optimisation locales ou globales ne peut avoir lieu qu'une fois on a précisé le problème traité, autrement dit la fonction objectif. Selon [51] l'optimisation est non seulement une théorie mathématique mais aussi une sorte de *cuisine algorithmique* où c'est principalement l'expérience qui guide l'utilisateur dans le choix de l'algorithme à implanter. Pour choisir

la méthode la plus adaptée à un problème bien précis, les caractéristiques principales prises en compte sont :

La capacité à éviter les minima locaux. C'est selon la complexité du problème qu'on peut choisir une ou l'autre des méthodes présentées en haut. Par exemple si la fonction objectif est convexe il est sans doute recommander de travailler avec une méthode locale qui permet d'atteindre l'optimum rapidement par rapport aux méthodes d'optimisation globale. Ceci dit, si la fonction objectif est multimodale ce n'est sans doute pas la peine d'appliquer une méthode local et le recourt à des méthodes globales devient nécessaire. Quoique là encore plusieurs méthodes existent, dans le présent manuscrit on s'est plutôt intéressé aux méthodes hybrides, vu leurs efficacité et efficience.

La robustesse d'un optimum. La robustesse d'un optimum est aussi une notion déterminante pour les concepteurs, et qui ne doit pas être négligée lors du choix de l'algorithme d'optimisation. La forme à optimiser est modélisée de manière déterministe ; cette forme est donc optimisée en négligeant les incertitudes liées par exemple aux procédés de fabrication ou aux simplifications du problème. Pour diminuer la probabilité que les performances de la forme optimisée ne soient pas vérifiées en réalité, on doit chercher un optimum robuste. Un optimum robuste est une solution peu sensible aux incertitudes.

La capacité à traiter des problèmes mono- ou multi- objectif. L'optimisation à objectifs multiples est un domaine de recherche très actif car les enjeux économiques et industriels sont énormes. Les méthodes d'optimisation multiobjectif, permettant de fournir au concepteur un ensemble de solutions correspondant à autant de compromis entre les différents objectifs antagonistes du problème. D'où leur importance.

La rapidité de convergence. C'est-à-dire quel est le nombre de variables à évaluer pour converger vers un optimum global ? La réponse à cette question réside dans le compromis qu'il faut trouver entre exploration et exploitation.

1.3.5 Exploration et exploitation des algorithmes d'optimisation

Pour un algorithme d'optimisation, **l'exploration** est sa capacité à explorer le domaine des variables pour rechercher la meilleure vallée, c'est à dire celle qui contient l'optimum global. A l'inverse, **l'exploitation** est sa capacité à converger rapidement vers le minimum d'une vallée donnée à partir d'un point de départ.

Le succès et l'efficacité d'une technique de résolution dépendent la

plupart du temps d'un compromis entre l'exploration et l'exploitation. Certaines méthodes toutefois n'utilisent qu'un seul de ces opérateurs pour parvenir à l'optimum. Ainsi, les méthodes déterministes, exploitant les dérivées de la fonction objectif et des contraintes pour atteindre rapidement et précisément le minimum local le plus proche du point de départ, privilégient l'exploitation au détriment de l'exploration.

Tout algorithme d'optimisation doit utiliser ces deux stratégies pour trouver l'optimum global : l'exploration pour la recherche de régions inexplorées de l'espace de recherche et l'exploitation pour exploiter la connaissance acquise aux points déjà visités et ainsi trouver des points meilleurs.

1.4 Optimisation continue avec contraintes

Toutes les méthodes décrites dans la section précédente s'appliquent à des problèmes d'optimisation sans contraintes. Or, en réalité il est quasiment impossible de trouver ce genre de problème, il est habituel de poser des contraintes sur les variables de conception ou encore des contraintes imposées par le cahier des charges. Donc, la prise en considération de ces contraintes lors de la résolution d'un problème d'optimisation doit avoir lieu. Autrement dit, certaines méthodes d'optimisation avec contraintes doivent être appliquées. L'idée est de substituer à la fonction à minimiser une autre fonction incluant les contraintes. On obtient alors l'optimum en cherchant les minima d'une suite de fonctions sans contraintes. Dans la suite on va donner une brève description des méthodes d'optimisation avec contraintes, les plus répandues.

1.4.1 Méthodes d'optimisation par transformation

Les méthodes d'optimisation par transformation consistent à introduire les contraintes de conception dans la fonction que l'on cherche à optimiser. Ainsi le problème avec les contraintes de type inégalité :

$$\begin{cases} Minimiser \;\; f(x) \\ Sous \;\; les \;\; contraintes \\ h_i(y) = 0, \quad i = 1, .., n \\ g_j(x) \leq 0, \quad j = 1, .., n \end{cases} \tag{1.19}$$

peut être transformé en un problème équivalent sans contraintes, formulé par :

$$\min \phi(x, r) = f(x) + r \sum_{j=1}^{n} P_j(g_j(x)) + \sum_{j=1}^{n} P_j(h_j(x))$$

où r est un paramètre (ou coefficient) de pénalité. La fonction de pénalité

P_j est construite de manière à favoriser la sélection des points admissibles par rapport aux points non admissibles, par la méthode d'optimisation sans contrainte qui sera utilisée. Les méthodes de pénalité sont divisées en deux classes selon la nature de la fonction de pénalité :

Méthodes de pénalités intérieures et extérieures

Les méthodes de pénalité *intérieures* utilisent une fonction de pénalité construite de telle façon que la possibilité de réalisation soit garantie dans l'ensemble du processus d'optimisation. De telles méthodes sont également appelées **méthodes à barrière** car la fonction de pénalité forme une barrière infinie le long de la frontière du domaine réalisable.

La fonction inverse : $P_j(g_j(x)) = \frac{-1}{g_j(x)}$

La fonction logarithmique : $P_j(g_j(x)) = -\log(-g_j(x))$

L'utilisation des méthodes à barrière impose que les paramètres soient situés dans l'espace réalisable. La recherche de points admissibles peut être effectuée à l'aide d'une méthode stochastique par échantillonnage aléatoire mais celle-ci peut conduire à des temps de calcul prohibitifs lorsque les contraintes sont sévères et que l'espace réalisable est très réduit par rapport à l'espace irréalisable. Ces méthodes à barrière, outre le fait qu'elles requièrent des jeux de paramètres réalisables, sont relativement difficiles à mettre en oeuvre.

Les méthodes de pénalités *extérieures* ne présentent pas d'inconvénient car elles sont valables aussi bien à l'intérieur qu'à l'extérieur de l'espace réalisable. Elles augmentent la pénalisation des solutions à mesure que l'on s'éloigne de l'espace réalisable. Les fonctions de pénalité extérieures les plus populaires sont les suivantes :

$$P_j(g_j(x)) = max[0, g_j(x)]$$
$$P_j(g_j(x)) = max[0, g_j(x)]^2$$

Le problème obtenu pourrait être résolu directement pour une valeur de r suffisamment grande de telle façon que les contraintes soient satisfaites mais ce choix entraîne un mauvais conditionnement de $\phi(x, r)$ et donc engendre un problème numérique lors de la résolution. Pour cette raison, les méthodes des pénalités sont en général résolues de manière itérative : une suite de valeurs croissantes de r est générée et à chaque itération k du processus, le problème d'optimisation sans contrainte suivant est résolu

$$\min \phi(x, r, k) = f(x) + r^k \sum_{j=1}^{n} P_j(g_j(x))$$

L'avantage de cette méthode est que le point de départ n'est pas nécessairement admissible tout en garantissant que le point final sera dans le domaine admissible ou presque. La fonction de pénalité extérieure est continue dans tout le domaine d'étude, admissible comme non admissible, mais elle présente l'inconvénient de conduire à un optimum réalisable seulement quand k tend vers l'infini et d'approcher ce point par une série de solutions non admissibles. Elle est utilisée pour les contraintes non uniquement fonction des variables d'optimisation ou les contraintes d'égalités [160].

Pénalisation radicale des solutions

La méthode de pénalisation radicale (ou **death penaly method**) est un processus de pénalisation très populaire dans le domaine de l'optimisation évolutionniste [168] . Il s'agit d'écarter les solutions non réalisables en attribuant à la fonction de transformation une valeur très élevée en cas de minimisation ($\phi \rightarrow \inf$) ou une valeur nulle en cas de maximisation ($\phi = 0$). La probabilité de survie de ces solutions, déterminée par les mécanismes de sélection, est alors quasi-nulle. Cette méthode est séduisante en raison de sa grande simplicité. Elle peut être appliquée avec succès lorsque l'espace de recherche est convexe et qu'il constitue une part importante de l'espace complet. Dans le cas contraire, cette approche a de sérieuses limitations, les solutions situées dans l'espace irréalisable ne pouvant être améliorées en raison de l'absence de directions données par la méthode de pénalisation.

1.4.2 Optimisation séquentielle des contraintes

Cette approche a été mise en oeuvre avec succès dans le domaine évolutionniste. Elle semble particulièrement bien adaptée à ces méthodes qui progressent à partir d'une population. La procédure consiste à utiliser à tour de rôle chaque contrainte comme fonction objectif d'un problème d'optimisation sans contrainte. Chaque contrainte devient successivement la fonction objectif alors que les autres sont ignorées. Ce processus de minimisation est interrompu dès qu'un pourcentage significatif de la population satisfait la contrainte. Lorsque toutes les contraintes ont été traitées, la fonction objectif est alors prise comme critère d'optimisation. La puissance de cette méthode réside dans la mémoire acquise par la population à chaque itération du processus. Bien que cette méthode paraisse coûteuse en temps de calcul, elle serait assez intéressante lorsque on est en présence de nombreuses contraintes difficiles à respecter [168, 202].

1.5 Optimisation multiobjectif

L'optimisation multiobjectif est un domaine fondamental de l'aide à la décision multi-critère, nécessaire aux nombreux milieux scientifiques et industriels. Au cours des deux dernières décennies, un très grand nombre de travaux, à la fois théoriques et appliqués, ont été publiés dans ce domaine. La résolution d'un problème d'optimisation multiobjectif consiste à déterminer la solution correspondante au mieux aux préférences du décideur parmi les solutions de bon compromis. L'une des questions les plus difficiles est donc liée à l'identification de l'ensemble Pareto optimal, ou d'une approximation de celui-ci pour des problèmes complexes. En particulier, beaucoup de problèmes de mécanique rencontrés dans l'industrie sont de nature multiobjectif.

Cette partie a pour but d'introduire les prérequis nécessaires à la bonne compréhension de cette thèse. Ainsi, dans un premier temps, nous présenterons le contexte de l'optimisation multiobjectif, les principales définitions, et les problématiques essentielles liées à ce domaine de recherche, surtout vis à vis des approches de résolution pour lesquelles nous allons essayer d'apporter un regard critique sur chacune d'elles.

1.5.1 Formulation d'un problème multiobjectif

Un problème multiobjectif ou multi-critère peut être défini comme un problème où on cherche à optimiser plusieurs composantes des vecteurs de la fonction objectif, toute en satisfaisant un ensemble de contraintes. Contrairement à un problème mono-objectif, la solution n'est pas unique mais constituée d'un ensemble de solutions dites de **Pareto optimales**.

La formulation mathématique d'un problème d'optimisation multiobjectif est donnée comme suit :

$$POM \quad \begin{cases} \text{Minimiser } F(x) = (f_1(x), ..., f_\ell(x))^T \\ \text{Sous les contrainte} \\ h_i(x) = 0 \quad , i = 1, ..., m, \\ g_j(x) \geq 0 \quad , j = 1, ..., k, \\ x_L \leq x \leq x_U \end{cases} \tag{1.20}$$

avec $x \in \mathbb{R}^d$ est le vecteur des variables de décision, $\{f_1, ..., f_\ell\}$ est l'ensemble des fonctions objectif.

La difficulté principale d'un problème multiobjectif est qu'il n'existe pas de définition de la solution optimale. Le décideur peut simplement exprimer le fait qu'une solution est préférable à une autre, mais il n'existe pas une solution meilleure que toutes les autres. Dès lors, résoudre un problème multiobjectif ne consiste pas à rechercher la solution optimale mais l'ensemble des solutions satisfaisantes pour lesquelles on ne pourra

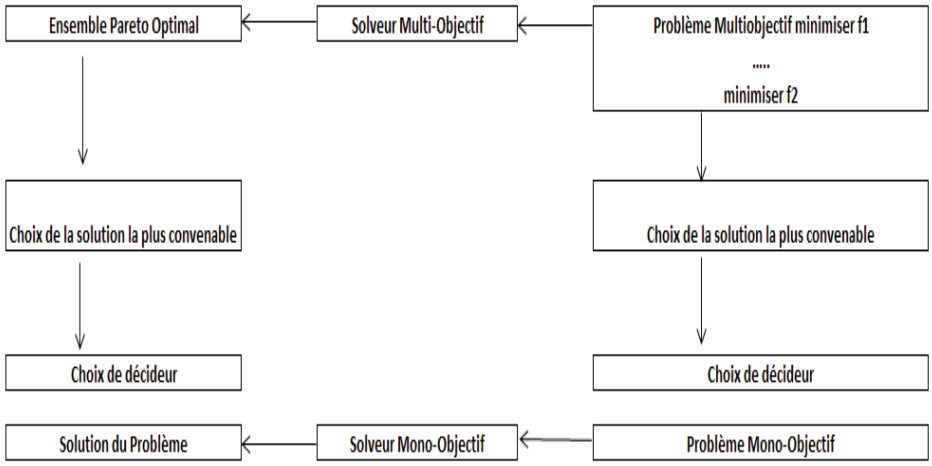

FIGURE 1.9 – Schéma des deux types de comportement en optimisation multiobjectif

pas effectuer une opération de classement. Les méthodes de résolution des problèmes multiobjectif sont donc des méthodes d'aide à la décision car le choix final sera laissé au décideur.

Pour répondre à ce problème, la communauté scientifique a adopté deux types de comportement.

• Ramener un problème multiobjectif à un problème mono-objectif en utilisant des pondérations ;

• Tenter d'apporter des réponses au problème en prenant en considération l'ensemble des fonctions objectif.

Il est très difficile de recommander à 100 % un comportement par rapport à l'autre. La différence entre ces deux comportements s'exprime dans la figure ci-dessous. Soit le décideur intervient dès le début de la définition du problème, en exprimant ses préférences, afin de transformer un problème multiobjectif en un problème mono-objectif. Soit le décideur effectue son choix dans l'ensemble des solutions proposées par la méthode d'optimisation multiobjectif.

La principale qualité d'une méthode d'optimisation multiobjectif est donc de rendre les décisions plus faciles et moins subjectives.

1.5.2 Classification des méthodes d'optimisation multiobjectif

Ce classement s'articule autour des notions de transformation et d'optimum de Pareto :

- Les méthodes agrégées ;
- Les méthodes fondées sur Pareto ;
- Les méthodes non agrégées et non Pareto.

Les méthodes agrégées ou l'utilisation de la dominance de Pareto traitent les objectifs simultanément, alors que, les méthodes dites non agrégées et non Pareto possèdent un processus de recherche qui traite séparément les objectifs.

Les méthodes agrégées

Ces méthodes transforment un problème multiobjectif en un problème mono-objectif. Pour ce faire on trouve dans la littérature deux modèles ; les modèles additif et multiplicatif. Cependant, les modèles les plus couramment utilisés sont le modèle additif

- **Les modèles additifs**.

$$U = \sum_{i=1}^{\ell} U_i(f_i)$$

U_i désigne la fonction de mise à l'échelle de la $i^{ème}$ fonction

- **Les modèles multiplicatifs**.

$$U = \prod_{i=1}^{\ell} U_i(f_i)$$

Les méthodes les plus répandues de cette classe sont :

1- La méthode de pondération, (weighting method)

Cette méthode consiste à additionner tous les objectifs en affectant à chacun d'eux un coefficient de pondération. Ce coefficient représente l'importance relative que le décideur attribue à l'objectif. Cela modifie un problème multiobjectif en un problème nono-objectif de la forme :

$$\min \sum_{i=1}^{\ell} w_i f_i(x) \quad \text{avec} \quad w_i \geq 0$$

w_i représente le poids affecté à la fonction objectif f_i avec $\sum_{i=1}^{\ell} w_i = 1$

Critique

Cette méthode est simple à mettre en oeuvre et elle est d'une grande efficacité. Mais la difficulté essentielle de cette approche est :

Comment le décideur peut déterminer les poids de chaque critère ?

Une solution à cette question est d'utiliser une combinaison linéaire des objectifs et de faire varier les poids de façon à constater l'influence de tel ou tel objectif sur le résultat, mais ça reste trop heuristique. Cette approche est facile à implémenter mais tous les résultats obtenus appartiennent à des zones convexes de l'espace des objectifs. Les solutions potentielles situées dans des portions concaves sont oubliées.

2- La méthode de programmation par but, (Goal programming)

Cette méthode est également appelée **target vector optimisation**[[48], [232]]. Le décideur fixe un but G_i à atteindre pour chaque fonction objectif f_i Ces valeurs sont ensuite ajoutées au problème comme des contraintes supplémentaires. La nouvelle fonction objectif est modifiée de façon à minimiser la somme des écarts entre les résultats et les buts à atteindre :

$$\min \sum_{i=1}^{\ell} |f_i(x) - G_i|$$

G_i représente la valeur à atteindre pour le $i^{ème}$ objectif.

Critique

Nous pouvons reprendre la critique faite pour la somme pondérée. La méthode est facile à mettre en oeuvre mais la définition des poids et des objectifs à atteindre est une question délicate qui détermine l'efficacité de la méthode.

3- La méthode du min-max

Cette méthode est assez proche de la précédente. Elle minimise le maximum de l'écart relatif entre un objectif et son but associé par le décideur.

$\min \max_i \Big(\frac{f_i(x) - G_i}{G_i} \Big)$ avec $i = 1, ..., \ell$
G_i représente le but à atteindre pour le $i^{ème}$ objectif.

4- La méthode du goal attainment

Dans cette approche le décideur spécifie l'ensemble des buts G_i, qu'il

souhaite atteindre et les poids associés w_i. La solution optimale est trouvée en résolvant le problème suivant :

$$\begin{cases} \text{Minimiser } \alpha \\ \text{Sous les contraintes} \\ G_i + \alpha w_i \geq f_i(x) \\ \sum_{i=0}^{\ell} w_i = 1 \end{cases}$$

Critique

Contrairement à la méthode de pondération, cette méthode peut générer des solutions sur la frontière de Pareto [4] en faisant varier les valeurs des w_i. Même dans le cas d'une surface concave citearticleChen94.

5- La méthode du ε contrainte

Cette méthode est basée sur la minimisation d'un objectif f_i en considérant que les autres objectifs $f_j \quad avec \quad j \neq i$ doivent être inférieurs à une valeur ε_j. En général, l'objectif choisi est celui que le décideur souhaite optimiser en priorité.

$$\begin{cases} \text{Minimiser } f_i(x) \\ \text{Sous les contraintes} \\ f_j(x) \leq \varepsilon_j, \ \forall \ j \neq i \end{cases}$$

De cette manière, un problème simple objectif sous contraintes peut être résolu. Le décideur peut ensuite réitérer ce processus sur un objectif différent jusqu'à ce qu'il trouve une solution satisfaisante. Cette méthode a été testée avec un algorithme génétique dans les travaux de [191] avec différentes valeurs de ε_j pour générer différentes solutions Pareto-optimales.

Critique

La connaissance a priori des intervalles appropriés pour les valeurs de ε_j est exigée pour tous les objectifs.

Les méthodes Pareto

L'idée d'utiliser la dominance au sens de Pareto a été proposée par [106] pour résoudre les problèmes proposés par [196]. Il suggère d'utiliser le concept d'optimalité de Pareto pour respecter l'intégralité de chaque critère car il refuse de comparer a priori les valeurs de différents critères.

4. Cette notion sera introduite dans la section suivante.

Dans la suite, nous définissons tout d'abord la notion de dominance au sens de Pareto, la frontière de Pareto et nous présentons quelques méthodes évolutionnaires utilisant cette notion.
- La théorie

Au $XIX^{ième}$ siècle, Vilfredo Pareto, un mathématicien italien, formule le concept suivant[175] : *dans un problème multiobjectif, il existe un équilibre tel que l'on ne peut pas améliorer un critère sans détériorer au moins un des autres critères*.

Cette équilibre a été appelé **optimum de Pareto**. Un point x est dit **Pareto-optimal** s'il n'est dominé par aucun autre point appartenant à C. Ces points sont également appelés solutions **non inférieures** ou **non dominées**.

La notion de dominance et frontière de Pareto

y domine z ssi

$$\forall i \in \{1,..,\ell\},\ \ f_i(y) \leq f_i(z) \quad \text{et} \quad \exists k \in \{1,..,\ell\}\ /\ f_k(y) < f_k(z)$$

On définit la frontière de Pareto comme étant l'ensemble de tous les points Pareto-optimaux.
- Les méthodes

Il existe de nombreuses méthodes Pareto, seules les principales sont listées ci dessous . Elles sont toutes basées sur des théories évolutionnaires.

a) Algorithme Génétique à plusieur objectifs : Multiple Objective Genetic Algorithm (MOGA)

En 1993 Fonseca [93] a proposé une méthode dans laquelle chaque individu de la population est rangé en fonction du nombre d'individus qui le dominent. Ensuite, ils utilisent une fonction de notation permettant de prendre en compte le rang de l'individu et le nombre d'individus ayant même rang.

Soit un individu x_i à la génération t, dominé par $p_i(t)$ individus. Le rang de cet individu est :

$$rang(x_i, t) = 1 + p_i(t)$$

Tous les individus non dominés sont de rang 1. Ils calculent la fitness de chaque individu de la façon suivante :

- Calcul du rang de chaque individu.

- Affectation de la fitness de chaque individu par application d'une

fonction de changement d'échelle sur la valeur de son rang.

L'utilisation de la sélection par rang a tendance à répartir la population autour d'un même optimum. Or cela n'est pas satisfaisant pour un décideur car cette méthode ne lui proposera qu'une seule solution. Pour éviter cette dérive, les auteurs utilisent une fonction de **sharing**. Ils espèrent ainsi répartir la population sur l'ensemble de la frontière de Pareto. Le sharing utilisé dans cette méthode agit sur l'espace des objectifs. Cela suppose que deux actions qui ont le même résultat dans l'espace des objectifs ne pourront pas être présentes dans la population.

D'une manière générale, cette méthode obtient des solutions de bonne qualité et son implémentation est facile. Mais les performances sont dépendantes de la valeur du paramètre σ_{share} utilisé dans le sharing. Dans leur article, Fonseca et Fleming [94] expliquent comment choisir au plus juste la valeur de σ_{share}.

b)Algorithme Génétique de tri non dominé : Nondominated Sorting Genetic Algorithm (NSGA II)

Srivinas ont implémenté en 1994 la première procédure de ranking qui a été initialement introduite par Goldberg [106], nommé Non Sorting Genetic Algorithm (NSGA). Tous les individus non-dominés de la population possèdent le rang 1. Ces individus sont ensuite enlevés de la population et l'ensemble suivant d'individus non-dominés est identifié et on leur attribue le rang 2. Ce processus est réitéré jusqu'à ce que tous les individus de la population aient un rang. Une nouvelle version de cet algorithme, nommée Non Sorting Genetic Algorithm II (NSGA II), a été présentée dans [68] . NSGA II intègre un opérateur de sélection, basé sur un calcul de la distance de "crowding" (ou "surpeuplement") [5] qui estime la densité de chaque individu dans la population.

Comparativement à NSGA , NSGA II obtient de meilleurs résultats sur toutes les instances présentées dans les travaux de K. Deb [67], ce qui fait de cet algorithme un des plus utilisés aujourd'hui.

c) Strength Pareto Evolutionary Algorithm (SPEA II)

L'algorithme SPEA II, proposé par Zitler [243], a été développé comme une amélioration de SPEA. SPEA est un algorithme qui maintient une population externe, appelée aussi archive, contenant le meilleur front de compromis rencontré durant la recherche. Cette population est destinée à contenir un nombre limité de solutions non-dominées trouvées depuis le début de l'exécution. A chaque itération, de nouvelles solutions non-dominées sont comparées aux individus de cette population au sens de

5. Le "crowding" fut introduit initialement par Jong .

dominance et seules les solutions non dominées résultantes restent. Il est à noter que SPEA, non seulement, préserve l'élite mais aussi la fait participer aux opérations génétiques. Une procédure de clustering est appliquée à cette population externe pour réduire sa taille si elle dépasse la taille limite.

SPEA II diffère de son prédécesseur en plusieurs aspects. D'abord, la taille de l'archive est fixe, c'est à dire que s'il n'y a pas suffisamment d'individus non-dominés, l'archive est complété par ceux dominés. Le calcul de performance est plus raffiné que celui de SPEA , au sens qu'il tient plus compte de la densité des solutions.

Les modifications apportées répondent à certaines critiques soulevées par SPEA. La procédure de clustering qui contrôlait la taille de l'archive dans SPEA est remplacée dans SPEA II par une méthode de troncation qui, contrairement au clustering, préserve les individus situés aux extrémités du front Pareto. Une autre différence importante est que dans SPEA II seuls les membres de l'archive participent au processus de reproduction.

d) Pareto Archived Evolution Strategy (PAES)

Cette méthode a été développée initialement comme une méthode de recherche locale dans un problème de routage d'information off-line. Les premiers travaux de Knowles et Corne [144] ont montré que cette méthode simple objectif fournissait des résultats supérieurs aux méthodes de recherche basées sur une population . Par conséquent, les auteurs ont adapté cette méthode aux problèmes multiobjectif. Les particularités de cette méthode sont les suivantes :

• Elle n'est pas basée sur une population. Elle n'utilise qu'un seul individu à la fois pour la recherche des solutions.

• Elle utilise une population annexe de taille déterminée permettant de stocker les solutions Pareto-optimales temporairement.

• Elle utilise une technique de surpeuplement (crowding) basée sur un découpage en hypercubes de l'espace des objectifs.

L'algorithme de PAES se présente en trois parties :

1) Génération d'une solution candidate.

2) Fonction d'acceptation de la solution candidate.

3) Archivage des solutions non dominées.

Cette méthode est relativement simple à mettre en oeuvre. De plus,

n'étant pas basée sur un algorithme génétique, elle évite à l'utilisateur le réglage de tous les paramètres de celui-ci. Mais son efficacité va dépendre du choix d'un nouveau paramètre l : le paramètre de discrétisation de l'espace des objectifs.

Les méthodes non agrégées et non Pareto

En général, les méthodes dites non agrégées et non Pareto possèdent un processus de recherche qui traite séparément les objectifs.

Fourman en 1985 [95] ont proposé la **méthode lexicographique** dans laquelle les objectifs sont préalablement rangés par ordre d'importance par le décideur. Ensuite, l'optimum est obtenu en minimisant tout d'abord la fonction objectif la plus importante puis la deuxième et ainsi de suite.

Schaffer en 1985 [196] propose une extension d'un algorithme génétique simple pour la résolution d'un problème multiobjectif dénommée **Vector Evaluated Genetic Algorithm (VEGA)**. La seule différence avec un algorithme génétique simple est la manière dont s'effectue la sélection.

En 1997 Dans les travaux de Valenzuela [231] , un algorithme non générationnel est proposé, **A Non Generational Genetic Algorithm** . L'idée d'utiliser un algorithme non générationnel provient des systèmes de classifeurs. Ils pensent qu'un problème multiobjectif a des caractéristiques similaires à celles des systèmes de classifeurs, donc qu'une approche non générationnelle peut être plus performante qu'un algorithme générationnel.

Les algorithmes non-Pareto sont souvent simples et faciles à mettre en oeuvre. Mais, ces méthodes répartissent la population sur les extrémaux du front de Pareto. En effet, l'évaluation mono-objectif des sous-populations ne permet pas la comparaison des diverses solutions multiobjectif disponibles à chaque itération en vue d'en exploiter les caractéristiques.

Nous donnons à la fin ce chapitre un tabelau récapitulatif des méthodes d'optimisation et de leurs caractéristiques :

Méthode	Dérivées	Convergence	Caractéristique	
			Positives	Négatives
GA	Heuristique	Aléatoire	Robuste,capable de gérer plusieur parametres	Très gourmande en terme du temps
PSO	Heuristique	Aléatoire	Rapide	_
DE	Heuristique	Aléatoire	Robuste	_
Pénalité	Dépendante de la technique	Dépendante de la technique	Robuste,capable de gérer plusieur état limites	Choix arbitraire des pénalités lenteur numérique
SQP	Gradient	Quadratique	Très haute efficacité	Calcul heissien
GA_PSO	Heuristique	Garantie	Très haute efficacité	Gourmande en terme du temps
DE_PSO	Heuristique	Sûre	Très robuste	Gourmande en terme du temps

TABLE 1.1 – Tableau de méthodes d'optimisation et leurs caractéristiques

1.6 Conclusion

Nous avons présenté dans ce chapitre les problèmes d'optimisation "difficile", qu'ils soient mono-objectif ou multiobjectif. Une famille de méthodes de résolution de ces problèmes a été présentée : les métaheuristiques. Les métaheuristiques sont des méthodes stochastiques qui visent à résoudre un large panel de problèmes, sans pour autant que l'utilisateur ait à modifier leur structure. Ces méthodes sont inspirées d'analogies avec des domaines aussi variés que la physique, la génétique ou encore l'éthologie. Les métaheuristiques ont vite rencontré un vif succès grâce à leur simplicité d'emploi mais aussi à leur forte modularité. On a vu tout au long de ce chapitre que, suivant la nature du problème posé (continu/discret, mono-objectif/multiobjectif, etc.), les métaheuristiques sont facilement adaptables et/ou hybridables en vue d'obtenir les meilleures performances possibles. Nous avons tiré que la richesse des différentes méthodes d'optimisation nous mène à choisir le bon compromis de méthodes et algorithmes afin de résoudre un problème.

Chapitre 2

Les Métaheuristiques

2.1 Introduction

Les "métaheuristiques" d'optimisation sont des algorithmes généraux d'optimisation applicables à une grande variété de problèmes. Elles sont apparues à partir des années 80, dans le but de résoudre au mieux des problèmes d'optimisation. Dans la pratique, trois types de problèmes d'optimisation sont souvent rencontrés : les problèmes combinatoires (discrets), les problèmes continus (à variables continues), et les problèmes mixtes [118]. L'exemple le plus répandu en optimisation combinatoire est celui du voyageur de commerce. En optimisation continue, un exemple simple est celui de la recherche des paramètres d'un modèle numérique pour approcher au mieux des données réelles[38]. Le dernier type est celui des problèmes mixtes, qui comportent à la fois des variables discrètes et des variables continues. Les métaheuristiques s'efforcent de résoudre tout type de problème d'optimisation. Elles sont caractérisées par leur caractère stochastique, ainsi que par leur origine discrète. Elles sont inspirées par des analogies avec la physique (recuit simulé, recuit microcanonique), avec la biologie (algorithmes évolutionnaires) ou encore l'éthologie (colonies de fourmis, essaims particulaires). Cependant, elles ont l'inconvénient d'avoir plusieurs paramètres à régler. Il est à souligner que les métaheuristiques se prêtent à toutes sortes d'extensions, notamment en optimisation mono objectif et multiobjectif. De nos jours les gestionnaires et les décideurs sont confrontés quotidiennement à des problèmes de complexité grandissante, qui surgissent dans des secteurs très divers. Le problème à résoudre peut souvent s'exprimer sous la forme générale d'un problème d'optimisation, dans lequel on définit une ou plusieurs fonctions objectif que l'on cherche à minimiser ou à maximiser par rapport à tous les paramètres concernés [111]. La résolution d'un tel problème a conduit les chercheurs à proposer des méthodes de plus en plus performantes, parmi lesquelles on trouve les métaheuristiques qui sont des méthodes générales de recherche dé-

diées aux problèmes d'optimisation difficile.
Dans ce chapitre nous présentons en premier lieu, d'une manière détaillée, les métaheuristiques les plus utilisées et nous mettons l'accent sur celles que nous avons étudiées dans le cadre de cette thèse. Ce sont les algorithmes génétiques, la méthode de l'évolution différentielle et la méthode de l'optimisation par essaim particulaire.
En second lieu nous présentons nos méthodes proposées avec des tests de leur validation et l'étude de leur comparaison avec d'autres méthodes.

2.2 Les métaheuristiques les plus utilisées

Reconnues depuis de nombreuses années pour leur efficacité, les métaheuristiques sont une famille de méthodes stochastiques qui consistent à la résolution des problèmes d'optimisation. Elles exploitent généralement des processus aléatoires dans l'exploration de l'espace de recherche pour faire face à l'explosion combinatoire engendrée par l'utilisation de méthodes exactes.
Leur particularité réside dans le fait qu'elles sont adaptables à un grand nombre de problèmes sans changements majeurs dans leurs algorithmes, d'où le qualificatif "méta". L'un des avantages de celles-ci est leur capacité à optimiser un problème à partir d'un nombre minimal d'informations, cependant elles n'offrent aucune garantie quant à l'optimalité de la meilleure solution trouvée. Seule une approximation de l'optimum global est donnée. Les métaheuristiques sont des méthodes qui ont un comportement itératif, c'est-à-dire que le même schéma est reproduit un certain nombre de fois au cours de l'optimisation, et elles sont directes, dans le sens où elles ne font pas appel au calcul du gradient de la fonction. L'utilisateur est certes, demandeur de méthodes rapides et efficaces, mais il est aussi demandeur de méthodes simples d'utilisation. Un enjeu majeur des métaheuristiques est donc de faciliter le choix des méthodes et de simplifier leurs réglages, afin de les adapter au mieux aux problèmes posés. Les métaheuristiques sont en évolution permanente. De nombreuses méthodes sont proposées chaque année pour améliorer la résolution des problèmes les plus complexes. Du fait de cette activité permanente, un grand nombre de classes de métaheuristiques existe actuellement. Les méthodes les plus courantes sont le recuit simulé, la recherche tabou, les algorithmes de colonies de fourmis ou encore l'optimisation par essaim particulaire. La section suivante est dédiée à la présentation de ces méthodes.

2.2.1 Méthode de colonies de fourmis

Introduction

La méthode de colonies de fourmis est apparue d'une constatation simple : les insectes sociaux, et en particulier les fourmis, résolvent naturellement des problèmes complexes. Un tel comportement est possible car les fourmis communiquent entre elles de manière indirecte par le dépôt de substances chimiques, appelées phéromones, sur le sol. Ce type de communication indirecte est appelé stigmergie. En effet, si un obstacle est introduit sur le chemin des fourmis, ces dernières vont, après une phase de recherche, avoir tendance toutes à prendre le plus court chemin entre le nid et l'obstacle. Plus le taux de phéromone à un endroit donné est important, plus une fourmi va avoir une grande probabilité à être attirée par cette zone. Les fourmis qui sont arrivées le plus rapidement au nid en passant par la source de nourriture sont celles qui ont pris la branche la plus courte du trajet. Il en découle donc que la quantité de phéromones sur ce trajet est plus importante que sur le trajet le plus long. De ce fait, le plus court chemin a une probabilité plus grande d'être pris par les fourmis que les autres chemins, et il sera donc pris par toutes les fourmis.

FIGURE 2.1 – Schéma de fourmis.

Principe

Le premier algorithme qui s'inspire de cette analogie a été proposé en 1996 par Colorni, Dorigo et Maniezzo [74] . Le but initial de cet algorithme était de résoudre le problème du voyageur de commerce. L'Algorithme 5 (voir annexe A) présente la méthode proposée par ces auteurs. Si l'on considère un problème de voyageur de commerce à N villes, chaque fourmi k parcourt le graphe et construit un trajet de longueur $n = |N|$. Pour chaque fourmi, le trajet d'une ville i à une ville j dépend de :

- La liste des villes déjà visitées, qui définit les mouvements possibles à chaque pas, quand la fourmi k est sur la ville i : J_i^k ;
- L'inverse de la distance entre les villes $\eta_{ij} = \frac{1}{d_{ij}}$, appelée visibilité. Cette information est utilisée pour diriger les fourmis vers des villes proches et ainsi, éviter de trop longs déplacements ;
- La quantité de phéromone déposée sur l'arête reliant deux villes τ_{ij}, appelée intensité de la piste. Cette quantité définit l'attractivité d'une piste, et elle est modifiée après le passage d'une fourmi. C'est la pseudo-mémoire du système [76].

La règle de déplacement est la suivante

$$p_{ij}^k(t) = \begin{cases} \dfrac{(\tau_{ij}(t))^\alpha (\eta_{ij})^\beta}{\sum_{l \in J_i^k} (\tau_{il}(t))^\alpha (\eta_{il})^\beta)} & \text{si} \quad j \in J_i^k \\ \\ 0 & \text{si} \quad j \notin J_i^k \end{cases} \tag{2.1}$$

où α et β sont deux paramètres contrôlant l'importance relative de l'intensité et de la visibilité. Après un tour complet, chaque fourmi dépose une quantité de phéromone $\Delta\tau_{ij}^k(t)$ sur l'ensemble de son parcours. Cette quantité dépend de la qualité de la solution trouvée, et elle est définie par

$$\Delta\tau_{ij}^k(t) = \begin{cases} \dfrac{Q}{L^k(t)} & \text{si} \quad (i,j) \in T^k(t) \\ \\ 0 & \text{si} \quad (i,j) \notin T^k(t) \end{cases} \tag{2.2}$$

où $T^k(t)$ est le trajet effectué par la fourmi k à l'itération t, $L^k(t)$ est la longueur de $T^k(t)$ et Q est un paramètre fixé.
Enfin, il est nécessaire d'introduire un processus d'évaporation des phéromones. En effet, pour éviter de se faire piéger dans des solutions sous-optimales, il est nécessaire qu'une fourmi "oublie" les mauvaises solutions. La règle de mise à jour est donc.

$$\tau_{ij}(t+1) = (1-\rho).\tau_{ij}(t) + \Delta\tau_{ij}(t) \tag{2.3}$$

où $\Delta\tau_{ij}(t) = \sum_{k=1}^{m} \Delta\tau_{ij}^k(t)$ et m est le nombre de fourmis.
Depuis le développement de la première méthode basée sur l'analogie de colonies de fourmis, cette méthode a été étendue à la résolution de plusieurs problèmes d'optimisation, discrets et continus.

2.2.2 Méthode de recuit simulé

Introduction

L'origine de la méthode du recuit simulé provient de la métallurgie, où, pour atteindre les états de basse énergie d'un solide, on chauffe celui-ci jusqu'à des températures très élevées, après on le laisse refroidir lentement. Ce processus est appelé le recuit. La méthode du recuit simulé a été développé simultanément par Kirk en 1983 [140] et Cerny en 1985 [41].

Principe

Cette méthode repose sur l'algorithme de Metropolis en 1953 [167]. Cet algorithme nous permet de sortir des minima locaux avec une probabilité élevée si la température T est élevée, et de conserver les états les plus probables pour des très basses températures. L'algorithme de Metropolis permet d'échantillonner la fonction objectif par le biais d'une distribution de Boltzmann de paramètre T.
Pour mieux comprendre la méthode du recuit simulé, il y a plusieurs notions à définir telles que la probabilité de Boltzmann et le critère de Metropolis.

- **Probabilité de Boltzmann**
  La probabilité de Boltzmann, notée P_T mesure la probabilité de trouver un système dans une configuration i avec une énergie E_i, à une température T donnée, dans l'espace des configurations S. Elle est définie par l'expression suivante

$$P_T(X = i) = \frac{1}{Z(T)} \exp(\frac{-E_i}{KT}) \tag{2.4}$$

  où X est une variable stochastique qui désigne l'état actuel du solide, K est appelé la constante de Boltzmann et $Z(T)$ est une fonction appelée fonction de répartition définie par

$$Z(T) = \sum_{j=1}^{t} \exp(\frac{-E_j}{KT}) \tag{2.5}$$

  où t représente tous les états énergétiques possibles.
  Le facteur KT montre que lorsque la température est très élevée, tous les états sont à peu près équiprobables, équivalent à dire qu'un grand nombre de configurations sont accessibles. Au contraire quand la température est basse, les états à haute énergie deviennent peu probables par rapport à ceux de faible énergie. Pour simuler l'évolution d'un solide vers l'équilibre thermique pour une température T, Metropolis a proposé un critère appelé critère de Metropolis et

qui est dérivé de la probabilité de Boltzmann.

– **Critère de Metropolis**
Dans le contexte de l'optimisation par la méthode du recuit simulé, l'énergie est remplacée par la fonction objectif, ainsi l'obtention d'un solide à énergie minimum est équivalente à la recherche de l'optimum global de la fonction objectif. Cette recherche se fait par des explorations successives de différentes configurations. Après chaque passage d'une configuration X à une configuration Y, la variation de la fonction objectif est $\Delta f = f(Y) - f(X)$. La transformation est acceptée selon la probabilité $p(X, Y)$ telle que

$$p(X, Y) = \exp(\frac{-\Delta f}{T}) \tag{2.6}$$

Lorsque la variation Δf est négative ou nulle, l'exponentielle est supérieure ou égale à 1 et la nouvelle configuration est acceptée.
Si $\Delta f > 0, \quad p(X, Y)$ est comparée à un nombre aléatoire $uniformedans[0, 1]$
Si $rand < p(X, Y)$, la configuration Y est acceptée
Sinon elle est rejetée et une autre configuration est essayée.
Les configurations ayant une augmentation en Δf, c'est à dire une dégradation de la fonction objectif sont donc moins probables pour une température donnée, d'autant moins que la température est faible.

Le point crucial de l'algorithme est donc la loi de décroissance de la température. De nombreuses lois de décroissance différentes ont été proposées par Siarry [45].

La méthode du recuit simulé est devenue rapidement populaire, du fait de sa facilité d'adaptation à un grand nombre de problèmes et son efficacité. En revanche, cette méthode présente l'inconvénient de contenir un grand nombre de paramètres (température initiale, règle de décroissance de la température, durée des paliers de température, etc.) qui rendent les réglages de la méthode assez empiriques. Cependant, il existe des études qui s'attachent à régler de manière optimale les paramètres de cette méthode [229]. Un autre inconvénient majeur de cette méthode, il s'agit de sa lenteur. Pour résoudre ce problème, plusieurs méthodes de parallélisassion des calculs ont été introduites par Azencott en 1992 [18]. Bien qu'initialement crée pour fonctionner avec des variables discrètes, la méthode du recuit simulé possède des versions continues.

2.2.3 Recherche Tabou

Introduction

La méthode de recherche Tabou est une métaheuristique originalement développée par Glover en 1986 [103] spécifiquement pour des pro-

blèmes d'optimisation combinatoire et qui permet de trouver d'une manière flexible un compromis entre la qualité de la solution et le temps de calcul. Le principe de cette méthode est d'ajouter au processus de recherche une mémoire flexible qui permet de mener une recherche plus "intelligente" dans l'espace des solutions.

Principe

Comme la méthode du recuit simulé, la méthode de recherche Tabou fonctionne avec une seule configuration courante, qui est actualisée au cours des itérations successives. La nouveauté ici est que, pour éviter le risque de retour à une configuration déjà visitée, on tient à jour une liste de mouvements interdits, appelée liste tabou. Cette liste contient m mouvements ($t \rightarrow s$) qui sont les inverses des m derniers mouvements ($s \rightarrow t$) effectués. La méthode modélise ainsi une forme primaire de mémoire à court terme. Dans sa forme de base, la méthode de recherche tabou présente l'avantage de comporter moins de paramètres que la méthode de recuit simulé. Cependant, la méthode n'étant pas toujours performante, il est souvent approprié de lui ajouter des processus d'intensification et/ou de diversification, qui introduisent de nouveaux paramètres de contrôle [104]. Dans la recherche tabou avec des variables continues, deux concepts sont mis en jeu : le voisinage d'un point donné et un mouvement aléatoire dans le voisinage. Pour un point X et un pas donné p, le voisinage $V(X, p)$ est défini comme suit

$$V(X, p) = Y : |X - Y| \leq p \tag{2.7}$$

où le point Y est généré aléatoirement dans le voisinage $V(X, p)$ de X. De plus s'il répond aux contraintes, il est appelé mouvement aléatoire faisable, ce voisinage est noté V. La taille du voisinage est le nombre de points Y générés dans le voisinage V de X avec un pas donné p. La performance de la recherche tabou dépend de la taille du voisinage et la longueur de la liste tabou.

En partant d'une solution quelconque X appartenant à l'ensemble des solutions S, la recherche se déplace vers une solution Y située dans le voisinage V par exploration itérativement de l'espace des solutions. Au début, le vecteur pas couvre la totalité de l'espace de recherche puis il diminue à chaque itération.
Parmi tous les points générés dans le voisinage V de X, le meilleur est retenu et sera le centre du prochain voisinage. Ce point est celui qui améliore le plus la fonction objectif f, ou sinon celui qui la dégrade le moins. L'algorithme a besoin d'une mémoire qui conserve la trace des dernières solutions visitées pendant un moment donné. Ces solutions sont déclarées taboues, d'où le nom de la méthode. Elles sont stockées dans une liste de longueur L fixe appelée liste taboue. Une nouvelle solution n'est acceptée que si elle n'appartient pas à la liste taboue. Le retour vers des solutions

déjà explorées dépendant donc de la longueur de liste taboue. Elle sera d'autant plus difficile que cette dernière est longue. En conséquence, la recherche sera dirigée vers des régions non explorées.

2.3 Algorithmes génétiques

Les algorithmes génétiques (AG) sont des algorithmes de recherche inspirés des mécanismes de l'évolution naturelle des êtres vivants et de la génétique. Les premiers travaux ont été menés par Holland en 1975 [114] dans l'ouvrage "Adaptation of Natural and Artificial System" qui formalise les algorithmes génétiques dans le cadre de l'optimisation mathématique. La parution de l'ouvrage de David Goldberg en 1989 [106]décrivant l'utilisation de ces algorithmes pour la résolution des problèmes concrets, a permis de mieux faire connaître ces derniers dans la communauté scientifique et a marqué le début d'un nouvel intérêt pour cette technique d'optimisation. Ces méthodes s'attachent à simuler le processus de sélection naturelle dans un environnement défavorable en s'inspirant de la théorie de l'évolution proposée par Darwin en 1859 " Journal of Research in to the Geology and Natural History". Selon ces concepts, lorsqu'une population est soumise aux contraintes d'un milieu naturel, seuls les individus les mieux adaptés survivent et génèrent une descendance. Au cours des générations, la sélection naturelle permet l'apparition d'individus de mieux en mieux adaptés au milieu naturel, au détriment de ceux se montrant notoirement inadaptés, assurant ainsi la pérennité de l'espèce. Cette particularité de l'évolution naturelle : la capacité d'une population à explorer son environnement en parallèle et à recombiner les meilleurs individus entre eux, est empruntée par les algorithmes génétiques.

2.3.1 Principes : Définition et vocabulaire

Par analogie à la génétique, ces algorithmes cherchent à partir d'une population initiale les meilleurs individus. Pour constituer les *parents* appropriés donnant naissance à des meilleures descendances *Enfants*, parmi lesquelle seront tirée les solutions acceptables du problème traité.

Chaque problème d'optimisation est caractérisé par des variables de décision qui conditionnent les décisions à prendre, un objectif à satisfaire et des contraintes à respecter. Contrairement aux méthodes d'optimisation classiques, les algorithmes génétiques n'utilisent pas les variables mais leur associe un codage particulier, les variables de décision sont prises en compte à l'aide de caractères représentant une séquence de codes. Chaque variable de décisions est traduite sous forme d'un gène qui peut contenir un ou plusieurs codes, pouvant exprimer des caractères différents. La séquence de code représente un individu. En d'autre terme une solution potentielle connue sous le nom de chromosome. L'objectif du problème traité

est exprimé à l'aide d'une fonction qui permet d'évaluer les chances qu'un individu soit sélectionné ou non pas, afin de reproduire de nouvelles solution. Cette fonction est la fonction d'adaptation (fitness). Les contraintes sont prises en compte dans la fonction d'adaptation en pénalisant les individus qui violent les contraintes du problème. L'exploration de l'espace des solutions possibles s'articule sur deux mécanismes qui cherchent à générer de manière aléatoire de nouvelles solutions à partir de la population de départ. Ces mécanismes sont les opérateurs génétiques : le croisement et la mutation. Le mécanisme de sélection cherche à diriger l'exploration en déterminant les individus ayant la plus grande probabilité d'être choisis.

Afin d'expliciter le fonctionnement des algorithmes génétiques, nous présentons dans la suite de cette section les différentes étapes d'un algorithme génétique simple illustrées sur la Figure 2.2.

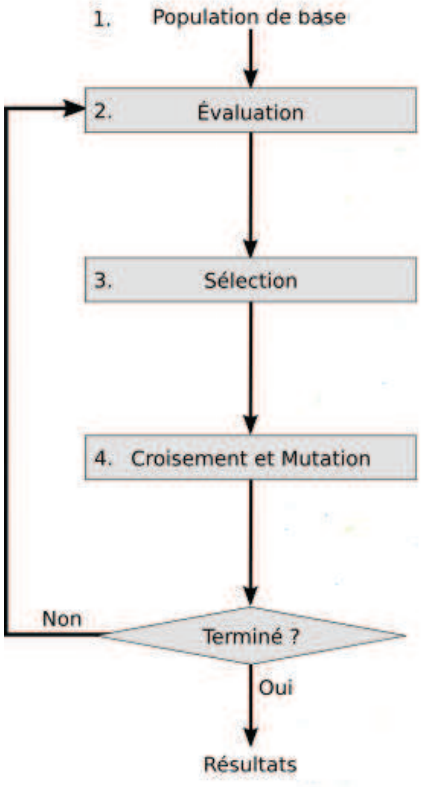

FIGURE 2.2 – Les principales étapes d'un algorithme génétique

2.3.2 Codage et population initiale

Un aspect important des algorithmes génétiques est la façon dont sont codées toutes les solutions. Les algorithmes génétiques établissent une analogie entre l'ensemble de solutions d'un problème et l'ensemble d'individus d'une population naturelle, en codant l'information sur chaque solution. En fonction du problème étudié, les codes utilisés peuvent être binaires ou réels.

L'AG démarre avec une population initiale d'individus dans le codage retenu. Le choix des individus conditionne fortement la rapidité de l'algorithme. Si la position de l'optimum dans l'espace de recherche est totalement inconnue, il est intéressant que la population soit répartie sur tout l'espace de recherche. Si par contre des informations à priori sur le problème sont disponibles, il paraît évident de générer les individus dans un espace particulier afin d'accélérer la convergence. Disposant d'une population initiale souvent non homogène, la diversité de la population doit être entretenue aux cours des générations afin d'explorer le plus largement possible l'espace de recherche.

2.3.3 Evaluation de la population

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

2.3.4 Opérateur de sélection

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

Sélection par roulette

Selon cette méthode, chaque chromosome est copié dans la nouvelle population proportionnellement à sa fitness. On effectue en quelque sorte, autant de tirages avec remise que d'éléments existant dans la population.

Ainsi pour un chromosome particulier ch_i de fitness $f(ch_i)$, la probabilité de sa sélection dans la nouvelle population de taille N est :

$$P\left(ch_i\right) = \frac{f\left(ch_i\right)}{\sum_{j=1}^{N} f\left(ch_j\right)} \tag{2.8}$$

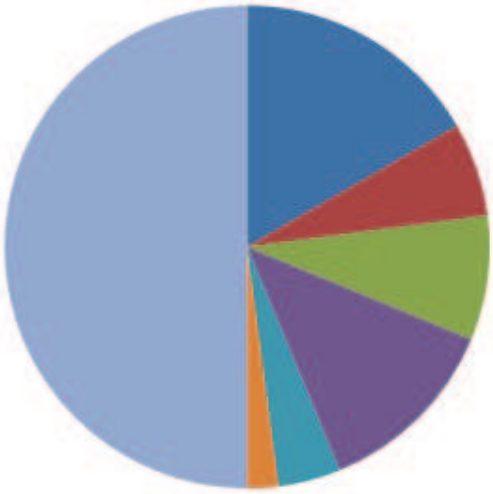

FIGURE 2.3 – séléction pa rouelette.

Sélection par tournoi

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

Sélection par rang

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

2.3.5 Opérateur de croisement

Le croisement est le principal opérateur agissant sur la population des parents. Il permet de créer de nouveaux individus par l'échange d'information entre les chromosomes par leur biais de leur combinaison. La population courante est divisée en deux sous populations de même taille et chaque couple formé par un membre provenant de chaque sous population participe à un croisement avec une probabilité souvent supérieure à 0.5. Si le croisement a eu lieu entre deux chromosomes parents (ch_1 et ch_2), on tire aléatoirement une position de chacun des parents. On échange ensuite les deux sous chaînes terminales de chacun des chromosomes, ce qui produit deux enfants ch'_1 et ch'_2 comme indiqué sur la figure 2.4.

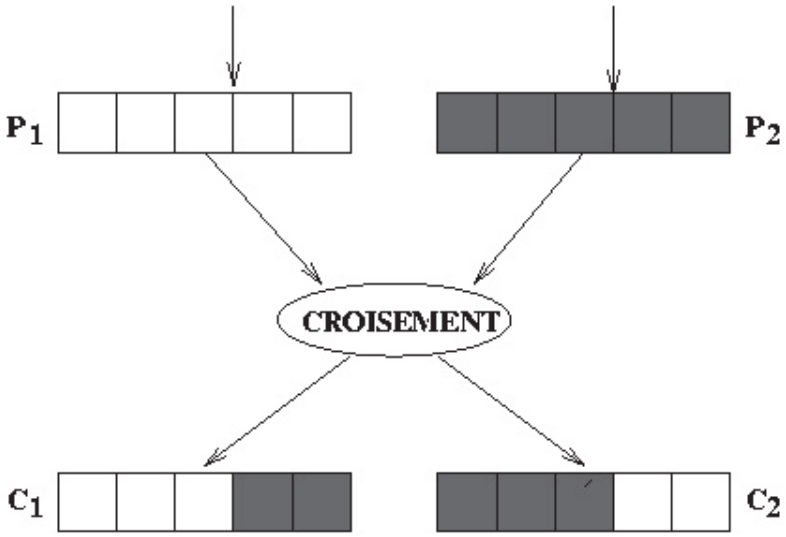

FIGURE 2.4 – Croisement standard en un seul point

Dans l'exemple de la figure 2.4, un croisement localisé à la quatrième position a eu lieu entre les chromosomes $ch1$ et $ch2$: il s'agit bien d'un croisement en un seul point. Ainsi on peut étendre ce principe de combinaison en choisissant non pas un seul point, mais 2, 3, etc... (croisement en multipoints). Sur la figure 2.5, un croisement en deux points est représenté.

Ce type de croisement est très efficace et peut s'étendre à n'importe quel type de chaînes (réelles, binaire, etc...). Néanmoins certains auteurs préfèrent utiliser dans le cas des chaînes réelles, un croisement de type barycentre : deux gènes $ch_1(i)$ et $ch_2(i)$ sont sélectionnés dans chacun des parents à la même position i. Ils définissent deux nouveaux gènes $ch'_1(i)$ et $ch'_2(i)$ par combinaison linéaire :

$$\begin{cases} ch'_1(i) = \alpha \times ch_1(i) + (1-\alpha) \times ch_2(i) \\ ch'_2(i) = (1-\alpha) \times ch_1(i) + \alpha \times ch_2(i) \end{cases} \tag{2.9}$$

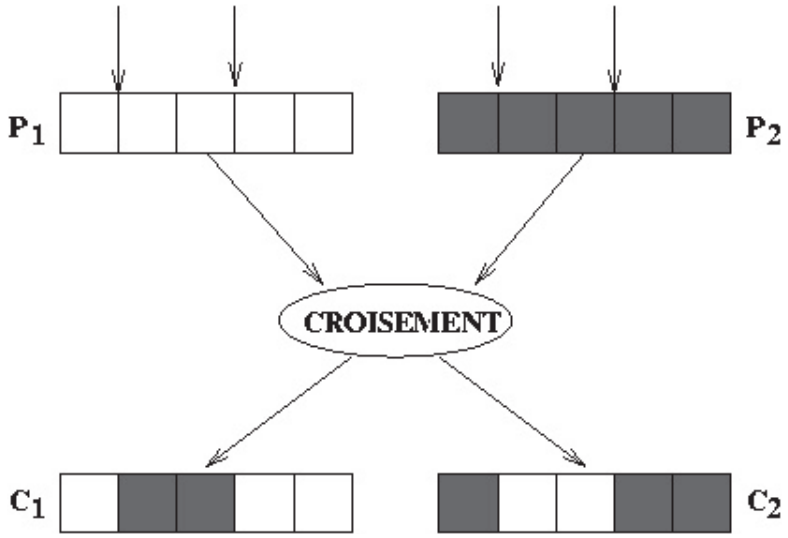

FIGURE 2.5 – Croisement standard en deux points

où α est un paramètre de pondération aléatoire qui prend généralement ses valeurs dans l'intervalle $[-0.5, 1.5]$ [50].

2.3.6 Opérateur de mutation

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

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

donnée par :

$$x'_k \begin{cases} x_k + \Delta\left(t, x_k^{\max} - x_k\right) & si\ rand = 0 \\ x_k - \Delta\left(t, x_k - x_k^{\min}\right) & si\ rand = 1 \end{cases} \tag{2.10}$$

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

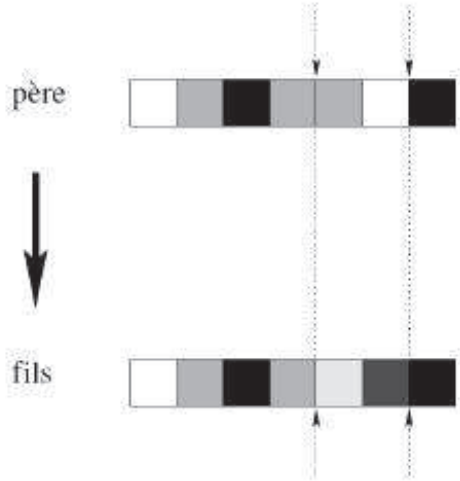

FIGURE 2.6 – Schéma de mutation

2.4 L'évolution différentielle

L'évolution différentielle ("Differential Evolution", DE) est une version améliorée des algorithmes génétiques, proposée par Price et Storn en 1995. DE est un algorithme basé sur une population initiale comme les algorithmes génétiques, il utilise les mêmes opérateurs : croisement, mutation et sélection. La différence principale en construisant de meilleures solutions est que les algorithmes génétiques se fondent sur le croisement tandis que le DE se fonde sur l'opération de mutation. Cette opération principale est basée sur la différence des paires de solutions aléatoirement tirées dans la population. L'algorithme utilise l'opération de mutation comme un mécanisme de recherche et l'opérateur de sélection pour diriger la convergence vers les régions éventuelles dans l'espace de recherche. DE utilise également un croisement non uniforme qui peut prendre des paramètres de vecteur d'enfant d'un seul parent.

DE démarre avec une population initiale de NP individus générée aléatoirement. Chaque individu est représenté par un vecteur de D dimension. A chaque génération, pour chaque individu $x_{i,g}$ un vecteur muté est créé selon la formule suivante :

$$v_{i,g+1} = x_{r1,g} + F.(x_{r2,g} - x_{r3,g}) \tag{2.11}$$

où les indices aléatoires $r_1, r_2, r_3 \in \{1, 2, ..., NP\}$ sont mutuellement différents et distincts de l'indice i. $F \in [0.5, 1]$ est le coefficient de mutation qui permet de contrôler l'amplitude des mutations. Un processus de croisement discret est introduit ensuite en vue d'augmenter la diversité de la population. Il combine le vecteur muté nouvellement créé avec le vecteur parent par la règle suivante :

$$u_{ji,g+1} = \begin{cases} v_{ji,g+1} \;\; si\ (rand_j \leq CR)\ ou\ (j = j_{rand}) \\ x_{ji,g} \;\; si\ (rand_j > CR)\ et\ (j \neq j_{rand}) \end{cases} \tag{2.12}$$

où $j = 1, 2, ..., D$; $\mathrm{rand_j} \in [0, 1]$; CR est la probabilité de croisement $\in [0, 1]$; et $rand_j$ est un indice aléatoire $\in (1, 2, ..., D)$ qui assure que $u_{i,g+1}$ ait au moins un paramètre issue du vecteur $v_{i,g+1}$. Finalement, l'opérateur de sélection compare la valeur de la fonction objectif des deux vecteurs $u_{i,g+1}$ et $x_{i,g}$ et le meilleur individu est sélectionné pour la population de la génération suivante. Ce processus se poursuit, génération après génération, jusqu'à atteindre le critère d'arrêt (nombre maximal de génération ou précision sur la valeur de la fonction objectif).

2.5 Optimisation par essaim particulaire

L'optimisation par essaim particulaire ("Particle Swarm Optimization", PSO) (Kennedy and Eberhart, 1995) [] est issue d'une analogie avec les comportements collectifs de déplacements d'animaux. En effet, chez certains groupes d'animaux, comme les bancs de poissons, on peut observer des dynamiques de déplacement relativement complexes, alors que les individus eux-mêmes n'ont accès qu'à des informations limitées, comme la position et la vitesse de leurs plus proches voisins. On peut par exemple observer qu'un banc de poissons est capable d'éviter un prédateur : d'abord en se divisant en deux groupes, puis en reformant le banc originel (voir figure 2.7), tout en maintenant la cohésion du banc. Ces comportements collectifs s'inscrivent tout à fait dans la théorie de l'auto-organisation. Pour résumer, chaque individu utilise, non seulement sa propre mémoire, mais aussi l'information locale sur ses plus proches voisins pour décider de son propre déplacement. Des règles simples, telles que " aller à la même vitesse que les autres ", " se déplacer dans la même direction " ou encore " rester proche de ses voisins " sont des exemples de comportements qui suffisent à maintenir la cohésion du groupe tout entier, et à permettre des

FIGURE 2.7 – Schéma de banc de poissons

comportements collectifs complexes et adaptatifs.

Kennedy et Eberhart se sont inspirés de ces comportements socio-psychologiques pour créer PSO. Un essaim de particules, qui sont des solutions potentielles au problème d'optimisation, se déplaçant sur l'espace de recherche pour trouver l'optimum global. Le déplacement d'une particule est influencé par les trois composantes suivantes :

- une composante physique : la particule tend à suivre sa direction courante de déplacement ;
- une composante cognitive : la particule tend à se diriger vers le meilleur site par lequel elle est déjà passée ;
- une composante sociale : la particule tend à se fier à l'expérience de ses congénères et, ainsi, à se diriger vers le meilleur site déjà atteint par ses voisins.

Dans le cas d'un problème d'optimisation, la qualité d'un site de l'espace de recherche est déterminée par la valeur de la fonction objectif en ce point. La Figure 2.8 illustre la stratégie de déplacement d'une particule.

Dans un espace de recherche de dimension n, chaque particule i de l'essaim est caractérisée par sa position $X_i = (x_{i1}, x_{i2}, ..., x_{in})$ et un vecteur de changement de position (appelé vélocité) $V_i = (v_{i1}, v_{i2}, ..., v_{in})$. chaque particule garde en mémoire la meilleure position par laquelle elle est déjà passée, que l'on note $P_i = (p_{i1}, p_{i2}, ..., p_{in})$. La meilleure position atteinte par toutes les particules de l'essaim est indiquée par un vecteur P_g. Avec ces notations, les équations de mouvement d'une particule sont, pour chaque itération $iter$:

$$V_i^{iter+1} = \chi \left(w V_i^{iter} + c_1 r_1^{iter} \left(P_i^{iter} - X_i^{iter} \right) + c_2 r_2^{iter} \left(P_g^{iter} - X_i^{iter} \right) \right) \tag{2.13}$$

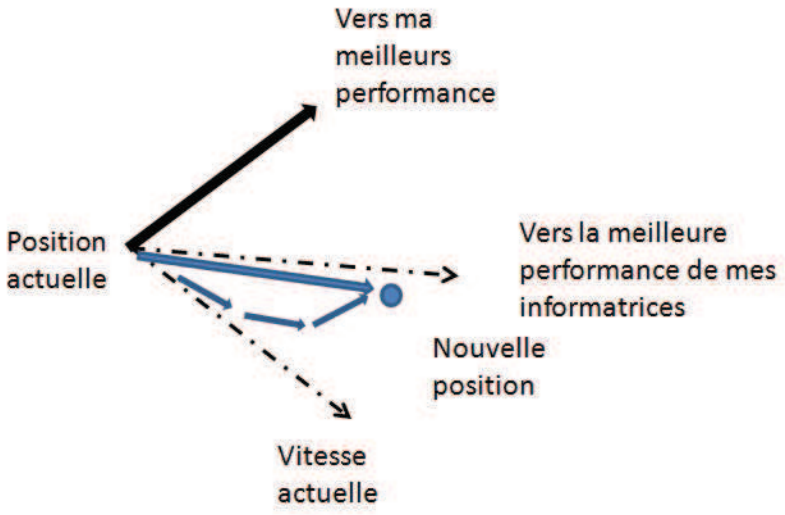

FIGURE 2.8 – Schéma de principe du déplacement d'une particule.

$$X_i^{iter+1} = X_i^{iter} + V_i^{iter+1} \tag{2.14}$$

où w est le coefficient d'inertie qui permet de régler la balance entre les phases diversification et intensification du processus de recherche. Les paramètres c_1 et c_2 contrôlent respectivement l'attraction à son meilleur et l'attraction au meilleur global. Enfin, r_1 et r_2 sont des variables aléatoires uniforme tirés dans $[0, 1]$.

PSO est un algorithme à population. Il commence par une initialisation aléatoire de l'essaim dans l'espace de recherche. A chaque itération de l'algorithme, chaque particule est déplacée suivant les équations (2.13) et (2.14). Une fois le déplacement des particules effectué, les nouvelles positions sont évaluées. Les P_i ainsi que P_g sont alors mis à jour. Cette procédure est résumée par l'Algorithme 3.1. N est le nombre de particules de l'essaim.

Des améliorations peuvent être apportées à cet algorithme de base, notamment du point de vue du contrôle de la divergence. Dans la version de base, il est possible que le déplacement d'une particule la conduise à sortir de l'espace de recherche. Dans ce cas, on peut assister à une amplification des rétroactions positives, qui conduit à une divergence de système. Pour s'affranchir de ce problème, (Eberhart et al., 1996) [132] ont introduit un nouveau paramètre $V_{\max}$ qui va permettre de contrôler l'éxplosion du système. Une étude sur le comportement de PSO suivant les valeurs de $V_{\max}$ est disponible dans (Fan et al., 2001). Dans (Clerc et al., 2002), il a été démontré qu'une bonne convergence peut être assurée en rendant dépendants les paramètres w, c_1 et c_2. L'utilisation d'un facteur de constriction

permet de prévenir l'explosion de l'essaim, d'assurer la convergence, mais aussi de s'affranchir de la définition arbitraire du paramètre $V_{\max}$.

Algorithme 3.1 : Algorithme d'optimisation par essaim particulaire.
Initialisation aléatoire des positions et des vitesses de chaque particule
Pour chaque particule i, $P_i = X_i$
Tant que le critèred'arrêt n'est pas atteint faire
  Pour $i = 1$ à N faire
    Déplacement de la particule à l'aide de (3.6) et (3.7)
    Évaluation des positions
    Si $f(X_i) < f(P_i)$
      $P_i = X_i$
    Fin Si
    Si $f(P_i) < f(P_g)$
      $P_g = P_i$
    Fin Si
Fin Pour
Fin Tant que

2.6 Conclusion

Dans ce chapitre, nous avons présenté quelques méthodes d'optimisation en s'appuyant sur les caractéristiques principales des métaheuristiques. Ces dérnières sont très efficaces en optimisation difficile sans avoir besoin de modifier la structure de base de l'algorithme utilisé. Elles sont devenues très populaires grâce à leur simplicité d'emploi dans différents domaines. Il est à noter qu'une bonne performance nécessite souvent une formalisation adéquate du problème posé et une adaptation intelligente d'une métaheuristique. Malgré le succès remarquable de leur démarche, les métaheuristiques présentent des difficultés à lesquelles est confronté l'utilisateur dans le cas d'un problème concret comme le choix d'une méthode efficace pour avoir une solution optimale et le réglage des paramètres qui peut être réalisable en théorie mais irréalisable en pratique. Les chercheurs visent à surpasser ces difficultés en proposant des techniques d'amélioration dont on cite l'hybridation des métaheuristiques. Cette hybridation exploite la puissance de plusieurs algorithmes et les combine en un seul méta-algorithme. Le chapitre suivant sera consacré à l'étude de l'hybridation de certaines métaheuristiques.

Chapitre 3

Hybridation et Méthodes proposées

3.1 Introduction

Ce chapitre est consacré pour la description des méthodes hybrides proposées pour réseoudre des problèmes d'optimisation et aussi à donner la notion de l'hybridation.
L'hybridation est une tendance observée dans de nombreux travaux réalisés sur les métaheuristiques ces dix dernières années. Elle permet de tirer profit des avantages cumulés de différentes métaheuristiques, à tel point que les métaheuristiques que nous avons vu jusqu'à présent ne sont plus que des canevas, des points de départ, pour commencer à résoudre un problème d'optimisation. L'hybridation consiste à combiner les caractéristiques de deux méthodes différentes pour tirer les avantages des deux méthodes. Les origines des algorithmes hybrides des métaheuristiques reviennent aux travaux de Glover [104]. Chacun d'eux a introduit une méthode de descente simple pour améliorer une recherche évolutive. Mais à cette période, la plupart des chercheurs n'y accordait que peu d'intérêt. Actuellement, les métaheuristiques hybrides sont devenues plus populaires car les meilleurs résultats trouvés pour plusieurs problèmes d'optimisation combinatoires ont étés obtenus avec des algorithmes hybrides. L'hybridation des métaheuristiques peut être divisée en deux grandes parties : hybridation des métaheuristiques avec des métaheuristiques et hybridation des métaheuristiques avec des méthodes exactes.

La question qui se pose est quel type d'hybridation est le plus performant. Pour notre cas avant de répondre a cette question ; nous proposons une classification de types d'hybridation.

3.2 Hybridation métaheuristiques/métaheuristiques

Toutes les méta heuristiques que nous avons étudiées précédemment dans le chapitre 2 ont été hybridées avec succès dans plusieurs applications.
Selon la taxonomie proposée dans [123] l'hybridation des métaheuristiques entre elles se fait en deux classifications principales. Une classification hiérarchique et une classification à plat .

3.2.1 Classification hiérarchique des métaheuristiques

Cette classification est caractérisée par le niveau et le mode de l'hybridation. Le niveau d'hybridation peut être bas (Low-Level) ou haut (High-Level). Dans le niveau bas, une métaheuristique remplace un opérateur d'une autre méthode qui l'englobe. Par contre, dans le niveau haut de l'hybridation, chaque métaheuristique garde sa propriété au cours de l'hybridation. Chaque niveau d'hybridation engendre deux modes de coopération à savoir, le mode relais et mode

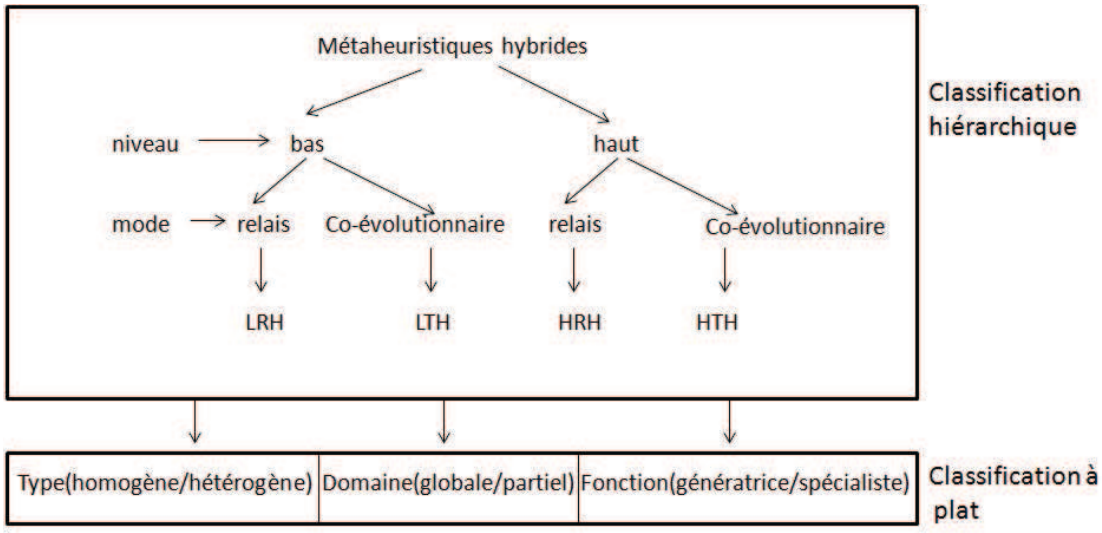

FIGURE 3.1 – Taxonomie de l'hybridation des métaheuristiques.

co-évolutionnaire. Dans le mode relais, les méthodes sont exécutées séquentiellement, c'est-à-dire le résultat de la première méthode est le début de la méthode suivante. Quand les différentes méthodes fonctionnent en parallèle pour explorer l'espace de recherche, on parle de mode co-évolutionnaire. La combinaison des modes et des niveaux donne quatre classes d'hybridation qui sont l'hybridation relais de bas niveau, l'hybridation co-évolutionnaire de bas niveau, l'hybridation relais de haut niveau

et l'hybridation co-évolutionnaire de haut niveau.

3.2.2 L'hybridation relais de bas niveau

Low-level Relay Hybrid. Elle englobe les métaheuristiques à base de solution unique dans lesquelles une autre méthode est incorporée pour former un nouvel algorithme. Pour résoudre le problème du voyageur de commerce (Traveling Salesman Problem) et le problème de la partition de graphe, Martin et Otto [162] ont inséré la méthode de descente dans un algorithme de recuit simulé.

3.2.3 L'hybridation co-évolutionnaire de bas niveau

Low-level Teamwork Hybrid. Consiste à incorporer une ou plusieurs métaheuristiques à base de solution unique dans une métaheuristique à population de solutions. L'avantage de ce type d'hybridation est de compenser la puissance d'exploitation d'une recherche locale et celle d'exploration d'une recherche globale. Stûtzle et Hoos [213] incorporent une fonction de recherche locale dans un algorithme de colonie de fourmis pour résoudre le problème du voyageur de commerce et celui de partition de graphes. Gambardella, Taillard [100] ont utilisé un algorithme de descente dans un algorithme de colonie de fourmis pour résoudre le problème de partition de graphes et McKendall et Shang [128] ont amélioré la méthode proposée par Gambardella, Taillard et al. en remplaçant l'algorithme de descente par l'algorithme du recuit simulé. Azimi [19] dans sa recherche de résolution du problème d'agencement d'horaire d'examens a proposé une amélioration de la solution de colonie de fourmis pour chaque cycle en appliquant le recuit simulé. Dans un algorithme génétique, J. Suh et D. van Gucht [110] ont remplacé l'opérateur de mutation par une recherche de descente et C. Fleurent et Ferland [83] ont appliqué cette classe d'hybridation pour résoudre le probléme de partition de graphe en remplaçant l'opérateur de mutation d'un algorithme génétique par une recherche tabou.

3.2.4 L'hybridation relais de haut niveau

High-level Relay Hybrid. Elle a lieu lorsque les métaheuristiques sont utilisées de manière séquentielle c'est-à-dire la (ou les) solution(s) finale(s) de la première métaheuristique est la (ou les) solution (s) initiale(s) de la métaheuristique suivante. Dans cette procédure, toutes les méthodes gardent leur intégrité. Talbi [217] ont introduit la recherche Tabou à la fin d'un algorithme génétique pour améliorer les solutions obtenues. Comme

il est connu d'utiliser une recherche locale pour initialiser une autre métaheuristique, Azimi a utilisé deux hybridations de type relais de haut niveau pour résoudre le probléme d'agencement d'horaires d'examens. Premièrement, il a initialisé les phéromones des colonies de fourmis par

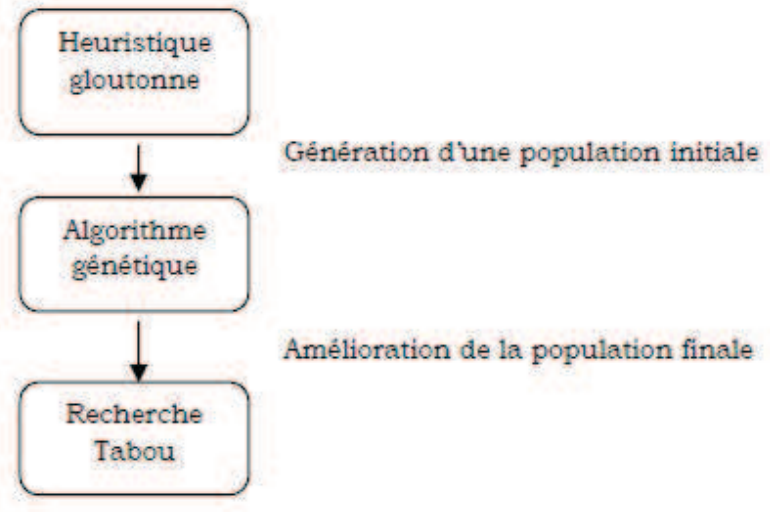

FIGURE 3.2 – Exemple d'une hybridation relais de haut niveau.

la recherche Tabou puis il a pris la solution trouvée par les algorithmes de colonie de fourmis comme une solution initiale d'une recherche Tabou. Pour résoudre le problème d'assignation quadratique, Lin, Kao et al. dans [156], proposent une hybridation où la méthode du recuit simulé crée une population initiale pour un algorithme génétique.
Dans le cas de l'hybridation co-évolutionnaire de haut niveau (High-level Teamwork Hybrid), les métaheuristiques utilisées travaillent en parallèle en échangeant des informations entre elles afin de trouver la solution optimale du problème posé. L'exemple le plus populaire est celui de l'algorithme génétique basé sur le modèle insulaire proposé par Tanese [52]. Pour cette hybridation, la population est divisée en sous-populations réparties sur les sommets d'un hypercube dont lesquels un algorithme génétique est lancé. Chaque sommet correspond à une zone de recherche de solutions et périodiquement, des individus migrent entre les sommets en contribuant à trouver les solutions optimales

Dans [33], Ghédira et Hammami ont fait appel à cette hybridation pour résoudre le problème de partitionnement de graphe. Ils ont utilisé la recherche Tabou et le recuit simulé de telle sorte qu'à chaque intervalle régulier, les deux méthodes échangent des informations pour trouver la meilleure solution. Notons qu'il existe d'autres travaux basés sur cette hybridation en utilisant des recuits simulés , des recherches tabou ,la programmation génétique et la stratégie évolutionnaire. Dans tous ces tra-

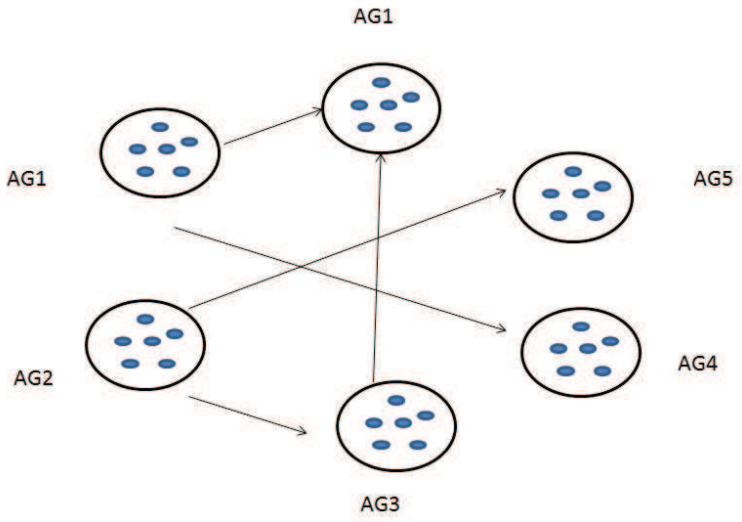

FIGURE 3.3 – Exemple de modèle insulaire des Algorithmes génétiques.

vaux, il a été démontré que les résultats ne sont plus performants si les métaheuristiques n'échangent pas d'informations.

3.2.5 Classification à plat des métaheuristiques

La classification à plat des métaheuristiques est caractérisée par le type des méthodes hybridées, leur domaine d'application et la nature de leurs fonctions. Selon le type d'hybridation, on trouve des méthodes hybridées homogènes où les algorithmes utilisés se basent sur la même métaheuristique comme le modèle insulaire et des méthodes hybridées hétérogènes où les métaheuristiques utilisées sont différentes. Le modèle proposé dans [52] est une hybridation de haut niveau co-évolutionnaire hétérogéne. Le domaine d'application des métaheuristiques hybridées permet de distinguer deux grandes classe d'hybridation, les hybridations globales et les hybridations partielles. L'hybridation globale a lieu lorsque toutes les méthodes hybridées sont appliquées à la totalité de l'espace de recherche. Toutes les méthodes que nous avons étudiées précédemment sont des hybridations globales. A l'opposé, l'hybridation partielle décompose un problème en sous-problèmes où chacun a son propre espace de recherche. Dans ce contexte, Taillard [20] propose une décomposition du problème de routage de véhicules en appliquant la recherche Tabou. L'idée est de diviser l'ensemble des villes à visiter en secteurs indépendants dont chacun représente un espace de recherche. Dans un autre article, Taillard et Voss [20] ont suggéré une méthode pour créer des sous- problèmes à partir du problème initial et le résoudre à l'aide d'une métaheuristique ou

d'une méthode exacte. Selon le problème traité, Talbi distingue deux types d'hybridation, une hybridation généraliste et une hybridation spécialiste. On parle d'hybridation généraliste quand toutes les métaheuristiques hybridées traitent le même problème d'optimisation. Toutes les hybridations que nous avons citées dans la classification hiérarchique font partie de cette catégorie. A l'inverse, les hybridations spécialistes ont lieu lorsque chaque métaheuristique traite un probléme différent. Un exemple de ce type est l'utilisation d'une métaheuristique pour initialiser les paramètres d'une autre métaheuristique. Krueger [146] optimise les paramètres d'un recuit simulé à l'aide d'un algorithme génétique. Abbattista [176] optimise ceux d'un algorithme de colonie de fourmis à l'aide d'un AG et Shahookar et Mazumder [42] optimisent les paramètres d'un algorithme génétique à l'aide d'un autre algorithme génétique.

3.3 Hybridation métaheuristiques/méthodes exactes

L'hybridation des métaheuristiques avec les méthodes exactes a été moins usuelle que l'hybridation métaheuristique/métaheuristique car la plupart des chercheurs la considérait assez inutile. En ces derniers temps, cette hybridation commence à s'étendre et un grand nombre d'articles a été publié concernant cette étude. Dans [219], Talbi a généralisé sa taxonomie aux méthodes exactes de sorte que la classification hiérarchique est applicable à ce genre d'hybridation. La classe d'hybridation relais de bas niveau (LRH) est plus efficace en ayant une métaheuristique hybridée avec une méthode exacte. Un exemple de ce type est proposé par Augerat [17] en utilisant un algorithme de Branch and Cut et une recherche tabou pour résoudre le problème de tournée de véhicule avec contraintes de capacité. La classe d'hybridation co-évolutionnaire de bas niveau (LTH) regroupe une métaheuristique à base de population dont laquelle un opérateur est remplacé par une méthode exacte. Cotta [55] proposent une hybridation entre un algorithme génétique et la méthode exacte Branch and Bound qui remplace l'opérateur de recombinaison. Pour résoudre le problème du voyageur de commerce, Jahuira propose des hybridations entre un algorithme génétique et une méthode exacte. Ici, l'opérateur de recombinaison est remplacé par un algorithme de Branch and Bound et la population initiale est choisie à l'aide d'un arbre de recouvrement minimal engendré d'une manière exacte. Une autre hybridation de cette classe a été proposée par Kostikas et Fragakis . Il s'agit d'implanter un algorithme de programmation génétique dans un algorithme de Branch and Bound pour générer la sélection des noeuds à explorer. La classe LTH concerne aussi toutes les métaheuristiques basées sur l'exploration du voisinage où une méthode exacte intervient pour trouver la (ou les) meilleure solution du voisinage. Bent et Van utilisent cette idée pour résoudre le pro-

bléme de voyageur de commerce. Dans la classe d'hybridation relais de haut niveau (HRH), les métaheuristiques et les méthodes exactes hybridées sont réalisées séquentiellement en gardant leur propriété. Bent et Van Hentenryck ont utilisé cette hybridation pour résoudre le problème de tourné de véhicule en minimisant le nombre de véhicules par le recuit simulé puis en optimisant la fonction objectif de chaque tournée par une recherche exacte dans un voisinage. Un autre exemple est la résolution du problème Flow-Shop proposée par Portmann [178]. L'hybridation co-évolutionnaire de haut niveau (HTH) est difficile à réaliser entre une méthode exacte et une métaheuristique car chaque approche résout un problème différent puis, un échange d'information est indispensable entre elles. Dans ce contexte, Chabrier [59] ont hybridé une recherche locale avec un algorithme de Branch and Price pour résoudre le problème de tournées de véhicules. L'exécution des méthodes s'effectue en parallèle tout en gardant une communication entre les méthodes. Pour la classification à plat, Talbi considére que les mêmes étapes qu'en hybridation métaheuristique/métaheuristique sont applicables à l'hybridation métaheuristique/méthodes exactes. Dans [28], Puchinger et Raid proposent une autre classification pour l'hybridation des métaheuristiques avec les méthodes exactes. Ils divisent les méthodes hybridées en deux grandes classes, les hybridations collaboratives et celle intégratives. Dans l'hybridation collaborative, les deux approches communiquent entre elles en échangeant des informations séquentiellement ou parallèlement. L'hybridation sequetielle a lieu quand une méthode exacte initialise une métaheuristique ou vice-versa. Applegate, Bixby [11] ont proposé une collaboration séquentielle pour résoudre le problème du voyageur de commerce en utilisant une recherche locale pour créer un ensemble de solutions puis un algorithme exact pour désigner la solution optimale. Dans [108], Klau, Ljubic et al. ont hybridé un algorithme mémétique avec la méthode Branch and Cut tel que le but de ce dernier est d'optimiser les solutions trouvées par l'algorithme mémétique. Feltl et Raidi ont utilisé la solution d'un programme linéaire d'une fonction pour sélectionner la population initiale d'un algorithme génétique. Pour résoudre le probléme d'ordonnancement de projet, Palpant, Artigues et al. [3] ont appliqué une métaheuristique pour construire la solution de départ puis un algorithme exact trouve une solution optimale. Applegate a utilisé cette stratégie pour résoudre les problèmes de Job-Shop. Quant à l'hybridation collaborative parallèle, elle a lieu quand la communication entre les métaheuristiques et les méthodes exactes s'effectue parallèlement. Pessan [226] présente une méthode d'hybridation d'un algorithme génétique avec la méthode Branch and Bound. L'idée est d'accélérer la résolution d'un problème par Branch and Bound et de réduire l'espace de recherche par l'algorithme génétique. La méthode la plus célébre de cette classe est celle créée par Talukdar [224] pour résoudre des problèmes de type asynchrone ou Asynchronous Teams (A-Teams). Cette méthode contient un ensemble d'agents et de mémoires

connectées à un réseau dirigé fortement cyclique. Chacun des agents est un algorithme d'optimisation approché ou exact qui peut travailler sur le probléme à résoudre. Une population d'individus est contenue dans la mémoire et les différents agents peuvent ajouter, détruire ou modifier des individus. Plusieurs travaux d'hybridations collaboratives parallèles sont issus de ce principe. Souza [4] a employé cette stratégie pour résoudre le problème du voyageur de commerce. Talukdar [224] l'ont utilisée pour le probléme de planification d'atelier pour le problème de planification d'horaires. Dans l'hybridation intégrative, l'un des algorithmes (métaheuristique ou exact) est intégré dans l'autre, c'est-à-dire qu'une métaheuristique est intégrée dans une méthode exacte ou inversement. Pour résoudre un probléme combinatoire de façon exacte, une méthode exacte est incorporée dans une métaheuristique. Burke, Cowling, dans [174], proposent une méthode de recherche locale incorporant un algorithmeexact pour résoudre le problème du voyageur de commerce. Raidl, Chu et Beasley présentent un algorithme génétique dont une méthode exacte est intégrée pour créer la population initiale ainsi que pour réparer les solutions non-réalisables. Congram et Potts, dans [1], proposent une méthode qui utilise la programmation dynamique dans un algorithme de recherche locale. Cotta et Troya [2] intégrent Branch and Bound dans un algorithme évolutif pour trouver le meilleur croisement possible entre deux individus sans avoir besoin d'appliquer une mutation, comme on trouve cette intégration pour décoder les chromosomes de certains algorithmes. Inversement, on peut intégrer une métaheuristique dans une méthode exacte. Khichane , remplace la procèdure de recherche arborescente par un algorithme de colonies de fourmis où chaque fourmi satisfait les contraintes du problème posé. Woodruff [238] dans son article intégre la recherche tabou dans Branch and Bound à chaque noeud de l'arbre pour trouver les meilleures solutions. Pour résoudre le problème de satisfiabilité maximale, French [97] ont proposé d'hybrider un algorithme génétique avec Branch and Bound. Au début, ils utilisent Branch and Bound pour créer la population initiale de l'algorithme génétique jusqu'à atteindre un certain critère et, à ce moment, l'algorithme génétique est lancé. Une fois terminé, les solutions trouvées sont insérées dans l'arbre de Branch and Bound qui reprend son exécution. Une autre hybridation entre les métaheuristiques et les méthodes exactes a été récemment proposée par Dumitrescu [77]. Cette classification n'est pas de type taxinomie. Elle fait la séparation selon l'objectif de la méthode utilisée pour l'hybridation. Dimistrescu et Stútzle proposent la classification suivante : -Utilisation d'une méthode exacte dans une recherche locale pour une exploration intensive du voisinage.

-Utilisation des solutions approchées de bonne qualité pour réduire l'espace de recherche d'une méthode exacte.

-Utilisation des bornes d'une méthode exacte pour une métaheuris-

tique à population.

-Utilisation des informations récoltées par une méthode exacte pour orienter un algorithme de recherche locale ou globale.

-Utilisation d'une méthode exacte pour une fonction spécifique de la métaheuristique.

A la fin de cet état d'art sur l'hybridation , celle ci est résumée en trois grand types :
-En parallèle.
-En série.
-En insertion.

3.4 Méthodes proposées

L'optimisation des structures est un processus essentiel dans la conception de systèmes d'ingénierie mécanique (génie civil, de l'aérospatiale, de l'automobile ...). Les industriels ne veulent pas seulement améliorer les performances mécaniques des pièces qu'ils conçoivent mais ils cherchent à optimiser leur poids, leur taille et également les coûts de production. En effet, pour résoudre ce type de problème, les algorithmes Meta-heuristiques robustes, qui nous permettent de réduire le coût de fabrication de la structure mécanique ainsi de maximiser le cycle de vie de la structure. Bien que les méthodes évolutives inappropriées sont plus difficiles à appliquer à des modèles mécaniques complexes en raison de calcul exponentielle du temps. Il est connu que les algorithmes DE ou GA sont très efficaces pour les problèmes difficiles, mais leur inconvénient est la consommation de temps. Comme ils sont très lourds et trop gourmand, Nous avons eu l'idée d'hybridation des deux algorithmes DE et GA par optimisation par essaim de particules. Cette expérience, a remarqué une grande amélioration, et elle est obtenue au niveau de la fonction objectif et aussi pour la minimisation de temps de calcul. Cependant, notre hybridation est une idée originale, car elle est un moyen de travail nouveau et différent de l'existant, nous expliquons l'avantage de l'hybridation. Ils sont généralement trois méthodes : hybridation en série , l'hybridation parallèle ou hybridation d'insertion. Nous avons opté pour l'hybridation d'insertion car elle est nouvelle et très efficace.

3.4.1 Hybridation de GA _ PSO

L'hybridation des algorithmes évolutionaires est une technique récente et bien efficace dans le cas des problèmes NP difficiles.

La technique d'hybridation adoptée dans notre travail est celle de l'hybridation en insertion décrtite par la suite.
Dans cette partie, nous allons discuter la structure de notre algorithme GA-PSO comme il est indiqué sur l'algorithme en dessous.
L'algorithme génétique comme nous avons décrit précédemment est formé par trois parties principales : la séléction, le croisement, et la mutation. Son inconvenient est qu'il est un peu lent quand la taille de la population devient grande. En effet ; l'idée d'hybridation en insertion est venue pour minimiser le temps de calcul et aussi pourquoi ne pas améliorer les résultats de la fonction objectif.
Cependant nous avons choisi d'hybrider les deux algorithmes heuristiques GA et PSO en insérant le PSO dans le GA. En d'autre terme à la place de faire la dernière étape de la mutation on la remplace par les mécanismes du PSO.
Par conséquent L'algorithme est récupéré en deux étapes :
Processus GA -Génération de la population initiale ;
-Séléction.
-Croisement.
Processus PSO -Recherche en voisinage de la meilleurs position.
-Injection des meilleurs particules et remplacer la mutation du GA.

Algorithme de GA_PSO

1- Méthode GA

Génerer une population initiale de taille N

Répeter

Pour $i = 1$ **Jusqu'à** N **Faire**

Evaluer la fonction coût de chacun des N individus.

Classer les sur la base de leurs fonctions coûts.

Sélection

Si $F(\overrightarrow{x}_i) > Fbest_i$ **Donc**

$Fbest_i = F(\overrightarrow{x}_i)$

Fin si

60 $\%$ **Croisement**

Pour Les meilleurs N individus, Appliquer le croisement de deux parents et les mettre à jour.

Les meilleurs N individus.

Fin pour.

Pour

A la place de GA-Mutation Nous appliquons la méthode PSO

Fin pour

2- Méthode PSO

Appliquer les opérateurs PSO (vitesse et position de mises à jour)

Pour la mise à jour des N individus atteints.

La mise à jour de vitesse et positions des particules

$V_{id}^{New} = w \times V_{id}^{Old} + c_1 \times rand \times (p_{id} - x_{id}^{Old}) + c_2 \times rand \times (p_{gd} - x_{id}^{Old})$ (1)

$x_{id}^{New} = x_{id}^{Old} + V_{id}^{New}$ (2)

avec $c_1 = c_2$ et $w = [0,5 + rand]$.

Jusqu'à ce que le critère d'arrêt soit atteint.

FIGURE 3.4 – Etapes d'algorithme hybride GA_PSO

3.4.2 Hybridation de DE _ PSO

L'hybridation des deux algorithmes DE et PSO se résume dans l'algorithme suivant :

Algorithme de DE_PSO

1. Génération de la population initiale. Le DE génère une population initiale des particules.

2. Évaluation de la fonction coût. Évaluer l'aptitude de chaque individu dans cette population.

3. Effectuer les opérateurs DE pour toutes les particules. Nous introduisons l'algorithme PSO pour obtenir cette équation :

Vi = g+ 1, G+ C (Pi, g - (Xi +1- Xi) / 2).

Avec *C = rand (C1, C2).*

4. Critères de licenciement. Nous nous arrêtons le processus lorsque l'optimum est obtenu.

Validation et comparaison avec d'autre méthodes

Dans cette section, nous présentons les résultats numériques obtenus par les algorithmes proposés GA_PSO et DE_PSO, pour des fonctions test. L'intérêt est de montrer que DE_PSO donne des résultats meilleurs pour la majorité des fonctions tests utilisées. Une variété des fonctions tests sont utilisées ; des fonctions unimodales (exemple : fonction sphère) et des fonctions multimodales (exemple : fonction de Rastrigin). Pour ces problèmes, la minimisation des fonctions tests choisies, sont sans contraintes mais avec des variables bornées.

	GA				PSO				GA_PSO			
Fonctions	**Fmin**	**Niter**	**temp/s**	**SR%**	**Fmin**	**Niter**	**temp/s**	**SR%**	**Fmin**	**Niter**	**temp/s**	**SR%**
Camel	-1.0316	171	0.241154	90	-1.0316	200	0.241154	98	-1.0316	13	0.0570735	98
Du joint	9.9770e-004	448	0.178250	85	4.7e-007	200	0.178250	85	2.1903e-008	34	0.01733710	86
Ratrigin	5.0473e-004	417	0.242241	80	0	200	0.242241	90	7.0985e-006	26	0.01302990	95
Shekel4_5	-10.1514	483	0.297276	40	-10.1532	200	0.297276	66	-10.1532	47	0.01223916	68
Shekel4_7	-10.4006	404	0.332279	30	-10.4029	200	0.332279	66	-10.4029	65	0.01132591	72
Shekel4_10	-10.5316	467	0.341914	30	-10.5364	200	0.341914	65	-10.5364	54	0.01756867	65
Shubert1	-12.8704	15	0.122546	90	-12.8709	200	0.122546	66	-12.8709	8	0.1018184	95
Shubert2	-186.7214	132	0.174803	90	-186.7309	200	0.174803	70	-186.7309	59	0.01277368	95
Shubert3	-2.7090e+003	451	0.172186	90	-2.7091+003	200	0.172186	66	-2.7008e+003	58	0.0139335	95
Shubert4	-3.9289e+004	431	0.178999	90	-2.2008+004	200	0.178999	54	-3.9300e+004	32	0.0381550	95

TABLE 3.1 – Comparaison des résultats

Function	GA	PSO	GA_PSO	DE	DE_PSO
Camel	-1.0316	-1.0316	-1.0316	-1.0315	-1.0316
Du joint	9.98E-04	4.70E-07	2.19E-08	2.19E-08	2.20E-08
Ratrigin	5.05E-04	0	7.10E-06	7.10E-06	7.05E-06
Shekel4_5	-10.1514	-10.1532	-10.1532	-10.1532	-10.1532
Shekel4_7	-10.4006	-10.4029	-10.4029	-10.403	-10.401
Shekel4_10	-10.5316	-10.5364	-10.5364	-10.536	-10.53
Shubert1	-12.8704	-12.8709	-12.8709	-12.8609	-12.8708
Shubert2	-186.7214	-186.731	-186.7309	-186.7309	-186.731
Shubert3	-2.71E+03	0.2909	-2.70E+03	-2.70E+04	-2.80E+04
Shubert4	-3.93E+04	1.7992	-3.93E+04	-3.95E+04	-3.40E+04

TABLE 3.2 – Comparaison de la fonction cout

Function	GA	PSO	GA_PSO	DE	DE_PSO
Camel	0.241154	0.241154	0.0570735	0.107345	0.0101335
Du joint	0.17825	0.17825	0.073371	0.13371	0.0102372
Ratrigin	0.242241	0.242241	0.030299	0.108931	0.0107831
Shekel4_5	0.297276	0.297276	0.0223916	0.246789	0.0206309
Shekel4_7	0.332279	0.332279	0.0132591	0.567892	0.0467002
Shekel4_10	0.341914	0.341914	0.0756867	0.300086	0.0200498
Shubert1	0.122546	0.122546	0.0018184	0.102344	0.0100694
Shubert2	0.174803	0.174803	0.0277368	0.12886	0.011142
Shubert3	0.172186	0.172186	0.0239335	0.123437	0.0122457
Shubert4	0.178999	0.178999	0.038155	0.13	0.012095

TABLE 3.3 – Comparaison de temps de calcul

Les tableaux de comparaison des résultats montrent que les méthodes hybrides sont meilleures ; la méthode DE_PSO est meilleure que la méthode GA_PSO.

3.4.3 Hybridation de GA _ PSO_NBI

Cette partie du chapitre est consacrée pour l'hybridation des algorithmes métaheuristique GA_PSO avec la méthode NBI. NBI a un certain nombre d'avantages par rapport aux autres méthodes existantes, y compris une garantie de la propagation de points dans l'ensemble de Pareto. NBI utilise une paramétrisation géométrique pour produire une répartition équilibrée de points sur le front de Pareto, en donnant une image précise de la surface totale. Il est généralement possible de trouver un en-

semble de poids tels que le point qui minimise une somme pondérée des objectifs.

De même, il est généralement possible de définir un problème de programmation pour lequel le point NBI est une solution optimale. NBI peut également traiter les problèmes où la surface de Pareto est discontinue ou non lisse, contrairement aux techniques d'homotopie. Malheureusement, un point produit par NBI ne peut pas être un Point de Pareto si la limite de l'ensemble obtenu dans l'espace contenant les points de Pareto est non convexe ou «plié» (ce qui arrive rarement dans les problèmes découlant de applications réelles). Dans cet section, nous proposons la méthode d'intersection de la frontière Normal (NBI) , qui a l'avantage de produire une répartition équilibrée des points Pareto. NBI fonctionne en transformant le problème d'optimisation multi-objectif en un ensemble de sous-problème de programmation non linéaire. NBI exige que les minimiseurs

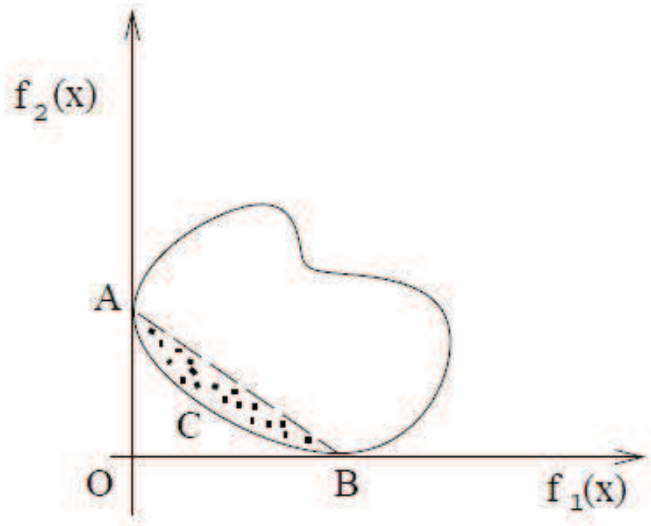

FIGURE 3.5 – Interpretation géometrique de NBI

individuels des différentes fonctions dès le départ, ce qui peut également être considéré comme un inconvénient. La formulation, cependant, doit être interprétée sous une autre façon. Au lieu d'une fonction objectif, nous avons des fonctions objectif que nous voulons minimiser sous contraintes. Puisque certaines des fonctions objectif peuvent entrer en conflit avec les autres, on doit trouver un compromis approprié en fonction des priorités de l'utilisateur. La situation idéale serait l'existence d'un vecteur x* avec

$$(f_1(\mathbf{x}^*), ..., f_\ell(\mathbf{x}^*)) = (f_1^*, ..., f_\ell^*)$$

dont chaque $f_i^*,\ \ i = 1, \cdots, n$, est la valeur minimale de l'individu corres-

pondant au problème scalaire

$$\begin{cases} \text{Minimiser } f_i(\mathbf{x}) \\ \text{sujet à } \mathbf{x} \in S \end{cases} \tag{3.1}$$

pour $i = 1, \cdots, n$. Mais bien sûr, c'est une situation idéale car normalement l'individu minimum f_i^* sera atteint aux différents points. NBI fonctionne essentiellement pour résoudre de manière séquentielle un ensemble problèmes non linéaires (NLP), qui sont définies comme suit :

$$\begin{cases} \text{Maximiser} \quad t \\ \text{sujet à} \quad \Phi w + t\hat{n} = F(\mathbf{x}) - f^*, \\ \mathbf{x} \in S \end{cases} \tag{3.2}$$

Φ la $\ell \times \ell$ la matrice dans laquelle i^e colonne on aF $(\mathbf{x}_i^*) - f^*$, où f^* est le vecteur contenant les minima individuelle des objectifs (par exemple, le point de l'utopie ou de l'ombreminimum) et $\mathbf{x}_i^*$ est le minimiseur des objectifs f_i. w est un vecteur des poids de telle sorte que $\sum_{i=1}^{\ell} w_i = 1, w_i \geq 0$ et $\hat{n}$ est le sens quasi-normalequi comporte des éléments négatifs, c'est à dire les points à l'origine. Φ w définit un point sur la convexe du Minima (CHIM). L'ensemble des points de $\mathbb{R}^n$ qui sont des combinaisons convexes de $f_i^* - f^*$, soit $\{\Phi w\}$, est dénommé CHIM.

Exemples numériques

Pour valider notre méthode hybride, nous proposons les exemples numériques suivant :

Exemple 1 : Ci-dessous nous avons un petit problème biobjective :

$$\begin{cases} \text{Minimiser}(f_1(x), f_2(x)) \\ f_1(x) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 \\ f_2(x) = 3x_1 + 2x_2 - x_3/3 + 0.01(x_4 - x_5)^3 \\ \text{Sous les contraintes :} \\ x_1 + 2x_2 - x_3 - 0.5x_4 + x_5 = 2 \\ 4x_1 - 2x_2 + 0.8x_3 + 0.6x_4 + 0.5x_5^2 = 0 \\ x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 \leq 10 \end{cases} \tag{3.3}$$

Le schéma suivant illustre le front de pareto qu'on a obtenu par la méthode GA_PSO_NBI.

Exemple 2 :

Pour évaluer la performance de l'approche GA-PSO-NBI pour résoudre des problèmes d'optimisation multi-objectif, un problème qui aborde les aspects de référence multimodaux fonctions objectifs et la notion d'opti-

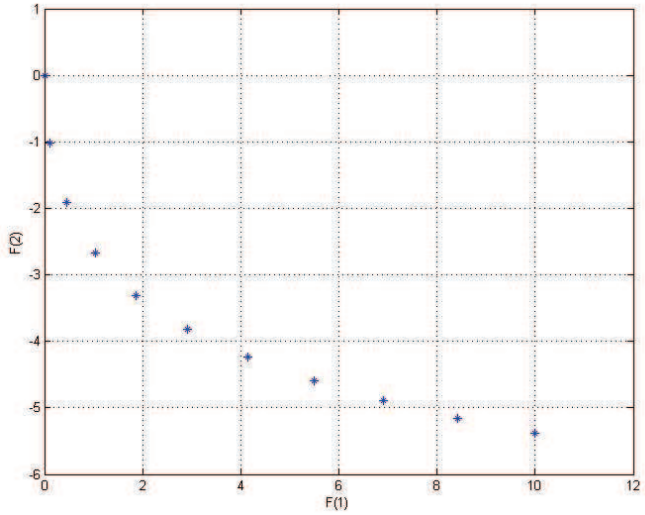

FIGURE 3.6 – Front de Pareto par la méthodes GA_PSO_NBI

mum de Pareto local est proposée. Ce problème a été choisi pour estimer la capacité de notre approche GA-PSO-NBI pour échapper à la convergence possible à une frontière de Pareto optimal local.

$$\begin{cases} \text{Minimiser} f_1(x_1, x_2), f_2(x_1, x_2 \\ f_1(x_1, x_2) = \sin(x_1 \Pi/2) \\ f_2(x_1, x_2) = ((1 - \exp(-(x_2 - 0.1)^2/0.0001)) + (1 - 0.5\exp(-(x_2 - 0.8)^2/0.8)))/\arctan(100x_1) \\ \text{Sous les contraintes :} \\ 0 \leq xi \leq 1; i = 1; 2 \end{cases} \tag{3.4}$$

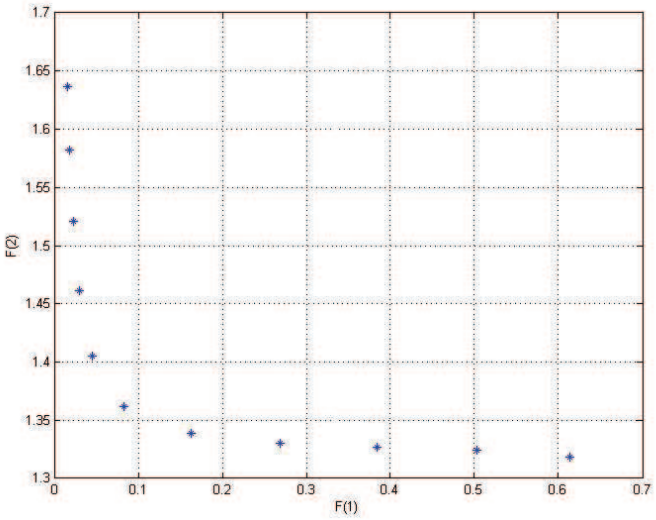

FIGURE 3.7 – Front de Pareto par la méthode GA_PSO_NBI

3.4.4 Hybridation de DE _ PSO_NNCM

Dans cette partie nous avons étudié les mêmes exemples avec notre méthode DE_PSO_NNCM et nous avons constaté qu'elle donne des résultats meilleurs que la méthode précédente.

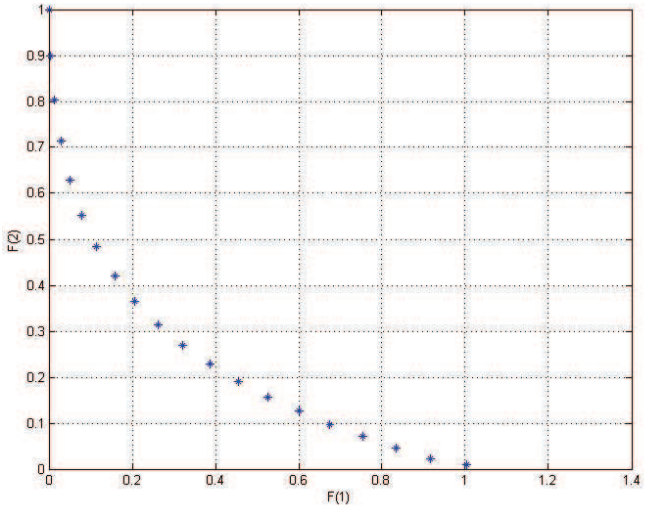

FIGURE 3.8 – Front de Pareto par la méthode DE_PSO_NNCM

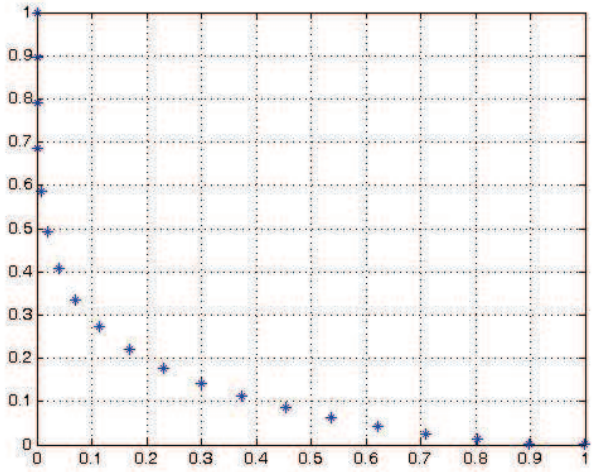

FIGURE 3.9 – Front de Pareto par la méthode DE_PSO_NNCM

3.5 Conclusion

Dans ce chapitre, nous avons présenté un diaporama sur les algorithmes évolutionnaires heuristiques.
Certes les méthodes sont trés variées mais les deux méthodes Hybrides que nous avons utilisées sont choisies vue leur rapidité et leur efficacité de résolution. Pour une recherche globale on a couplé le PSO en insertion au niveau des GA et en série au niveau de DE pour donner un souffle rapide ; qui minimisera par la suite le temps de calcul ainsi l'amélioration de la fonction coût (objectif).
Nous avons jugé que l'hybridation est la meilleure solution dans les cas d'optimisation des systèmes. Les tableaux de comparaison des résultats montrent que les méthodes hybrides sont plus efficaces et efficientes.

Chapitre 4

Applications en optimisation mono-objectif

4.1 Introduction

L'optimisation mono-objectif des structures soulève depuis plus de deux décennies le plus vif intérêt. Encore trop peu appliquée aux techniques classiques de bureaux d'études, elle s'y intègre progressivement au fur et à mesure que s'accroît sa robustesse. Parti des problèmes les plus simples, le champ d'application de l'optimisation structurale s'étend aujourd'hui à de nouveaux défis toujours plus intéressants [147]. La détermination de formes adéquates des composants structuraux est un problème de première importance pour l'ingénieur. Dans tous les domaines de la mécanique des structures, l'impact de la bonne conception d'une pièce est très important sur sa résistance, sa durée de vie et son utilisation en service. Ce défi est quotidien dans les secteurs de pointe tels que la recherche spatiale, électronique, l'aéronautique, l'automobile, la construction navale, la mécanique de précision et les ouvrages d'art en génie civil [136]. Le développement de l'art de l'ingénieur requiert des efforts considérables pour améliorer sans cesse les techniques de conception des structures. L'optimisation intervient de façon primordiale dans l'augmentation des performances soit de masse, de position ou de coût. Ce chapitre est consacré aux problèmes mono-objectifs dans le domaine de la mécatronique. Ces travaux abordent le problème des cartes électroniques, qui ont été traités dans le cadre de la collaboration avec les deux laboratoires LERMA et le ERD3M à l'EMI.

4.2 Optimisation des cartes électroniques

4.2.1 Optimisation de la position de la vis d'une carte électronique

Introduction

Les systèmes électroniques embarqués jouent un rôle très important dans plusieurs domaines, tels que l'industrie automobile, l'aéronautique, les télécommunications, le secteur médical... etc. Afin d'assurer leurs fonctionnements, les systèmes électroniques doivent être fiables. Toutefois, l'un des plus grands problèmes des systèmes mécatroniques embarqués concerne l'évaluation de leur fiabilité. Les systèmes mécatroniques sont sujets à une dégradation liée à des modes de défaillance multiples selon les technologies utilisées et les conditions d'utilisation. L'utilisation simultanée de plusieurs technologies augmente les risques de dysfonctionnement des systèmes mécatroniques. Afin d'obtenir des systèmes dans lesquels les utilisateurs placent une grande confiance, des études de sûreté de fonctionnement, et en particulier de fiabilité, doivent être menées tout au long du cycle de développement ou de vie du système : de la spécification jusqu'à la validation et à la mise en exploitation. L'objectif de ce travail est d'appliquer une nouvelle méthode hybride d'optimisation GA_PSO à une plaque électronique pour trouver la position optimale de sa cinquième vis de fixation qui assure sa fiabilité.

Modèle de simulation

La défaillance du circuit imprimé (PCB) est un point crucial pour assurer la fiabilité des cartes électroniques. Dans cette étude, la modélisation mécanique du PCB est réalisée par l'outil de calcul ANSYS et le code d'optimisation est implémenté sous Matlab. Ce travail vise d'une part l'analyse de la réponse du PCB pour une meilleure compréhension du comportement thermomécanique mis en jeu, et d'autre part la détermination de la position optimale de la vis qui minimise la déformation de la carte électronique.

Modélisation géométrique

Le modèle géométrique, Figure 4.1, est une carte électronique embarquée sous forme d'une plaque rectangulaire avec quatre couches de cuivre (CU) et trois couches de composite FR4. Elle est encastrée par cinq vis comme le montre la figure 4.2. Elle est soumise à une température uniforme T. Un modèle en éléments finis $2D$ est développé. Les propriétés de ces matériaux sont données au Tableau 4.1. L'intérêt de l'utilisation du modèle local réside dans la réduction du temps de calcul. Les conditions

aux limites sont introduites dans la modélisation.

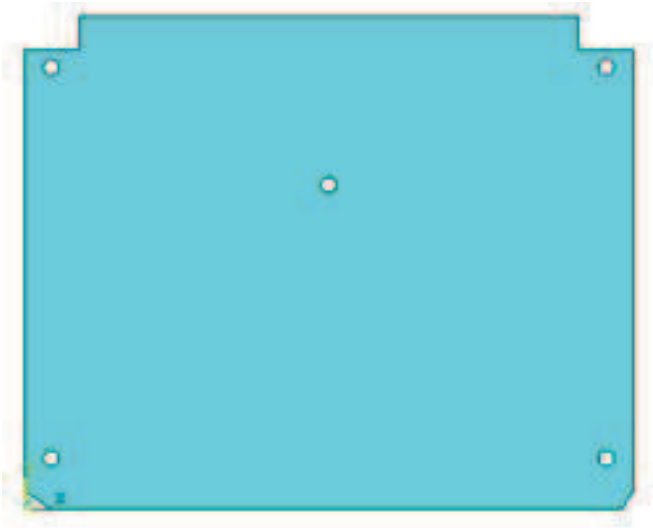

FIGURE 4.1 – Modèle géométrique

Maillage

La figure 4.2 présente le maillage utilisé pour la simulation thermomécanique de la carte électronique.

Propriétés des matériaux

Les propriétés des matériaux constituant la structure étudiée sont présentées dans le tableau 4.1.

Propriétés des matériaux	FR4	Cu
Module de Young (GPa)	17	115
Coefficient de Poisson	0.39	0.31
Densité (Kg/m3)	1800	8890
CTE (mm/K)	18	17
Module de cisaillement (GPa)	2.4	44

TABLE 4.1 – Propriétés des matériaux

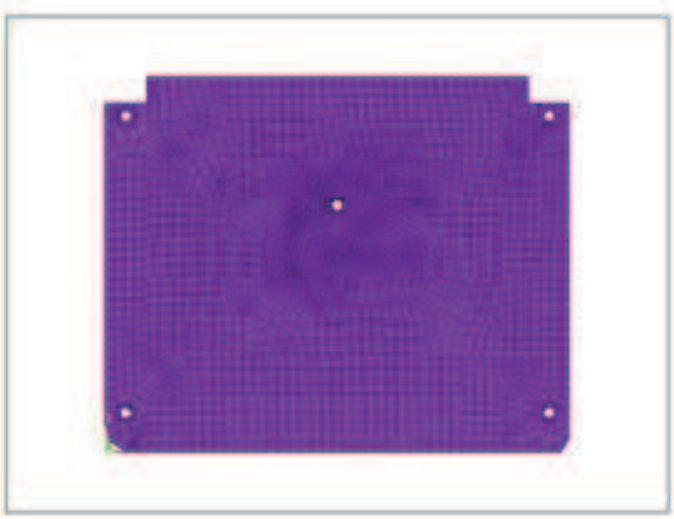

FIGURE 4.2 – Maillage

Conditions aux limites et chargement

La structure est encastrée par 4 vis aux 4 coins et par une cinquième vis avec une position qui change aléatoirement pour trouver la position optimale qui minimise la déformation du PCB. La structure est soumise à une température constante.

Calcul du déplacement maximal

Le calcul de la déformation du circuit imprimé en figure 4.3 indique la zone fortement déformée. La position de la cinquième vis du PCB est alors caractéristique puisque le déplacement maximal se produira autour cette zone. Une identification de cette position caractéristique a été réalisée afin d'alimenter le modèle d'optimisation.

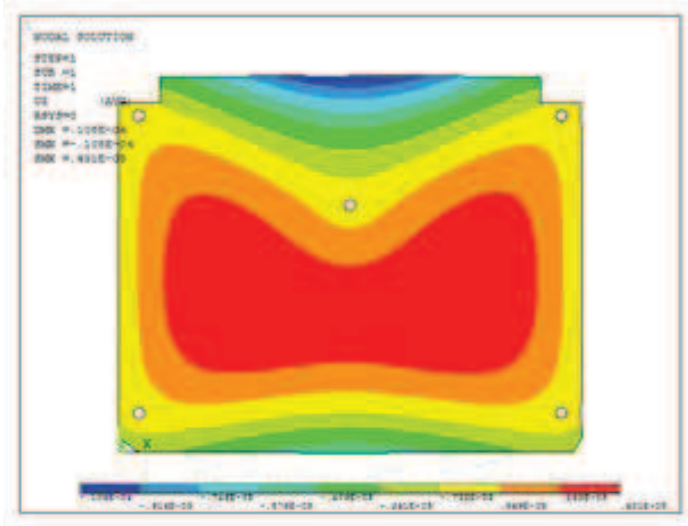

FIGURE 4.3 – Déplacement maximal

Modèle mathématique

Pour cette application, nous cherchons la position optimale de la cinquième vis de fixation de la plaque embarquée. La plaque est soumise à une température uniforme. Le déplacement maximal Dmax de la plaque est fixé.

$$\begin{cases} \text{Minimiser } f(x) = Dmax(x, y) \\ \text{sous les contraintes} \\ 10 \leq x \leq 160 \\ 15 \leq Y \leq 121 \end{cases} \tag{4.1}$$

la méthodologie adoptée pour réaliser le couplage entre matlab et Ansys pour trouver les paramètres géométriques optimaux de la plaque électronique étudiée est schématisée ci-dessous .

Résultats et discussion

Les résultats obtenus avec la méthode d'optimisation présentée ici sont comparés dans le tableau 4.2 avec les résultats déterministes. Nous notons que la méthode d'optimisation heuristique donne de bons résultats en comparaison avec la méthode déterministe et permet à notre carte électronique un bénéfice de déplacement de 0,003. Cet avantage permet une plus longue vie à la carte électronique et également pour les composants électroniques qui sont placés dans la carte électronique. La figure 4.5 montre le maillage de la carte électronique de la position optimale.

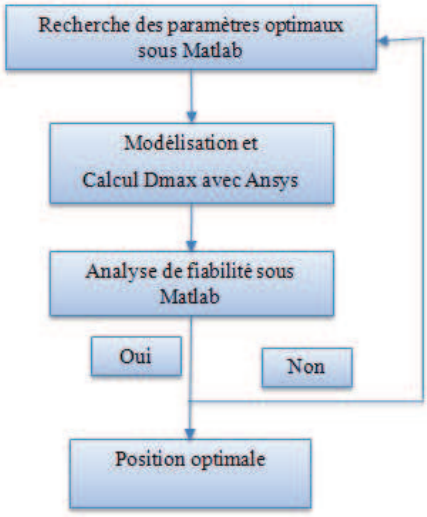

FIGURE 4.4 – Méthodologie de l'optimisation fiabiliste

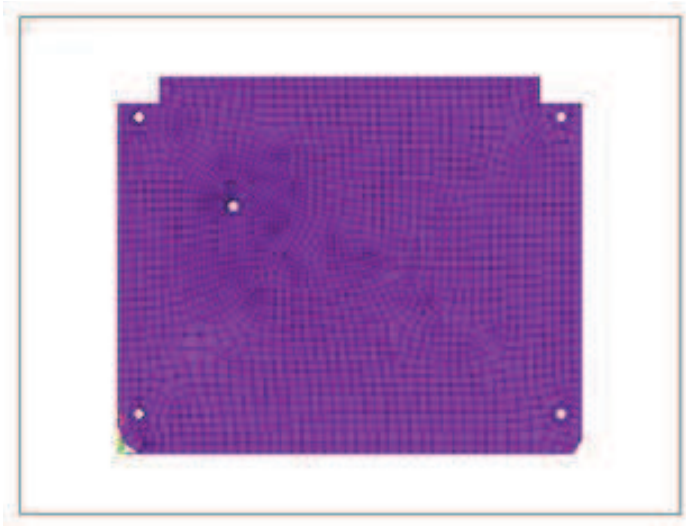

FIGURE 4.5 – Maillage avec la position optimale

	Résultats déterministes(mm)	Résultats évolutionnaires (mm)
x	84.25	41.68
y	89.75	89.78
Déplacement maximal	0.01	0.007

TABLE 4.2 – Comparaison des résultats déterministes et évolutionnaires

La figure 4.6 montre le déplacement maximal de la carte électronique avec la position optimale.

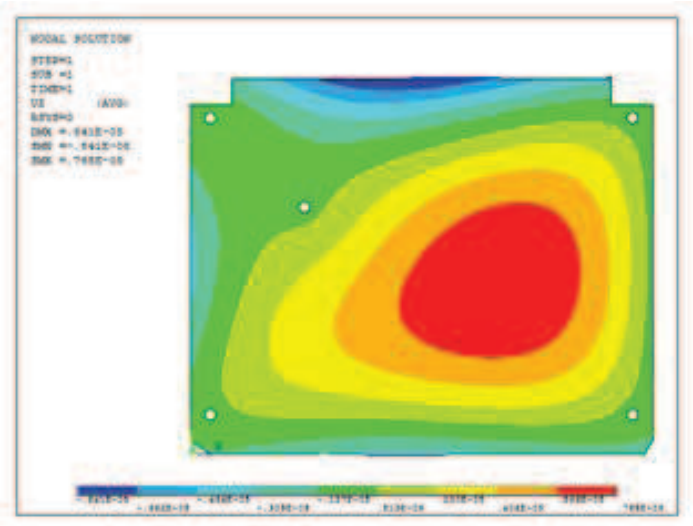

FIGURE 4.6 – Déplacement maximal avec la position optimale

Concluion

Dans ce travail, la modélisation mécanique de la carte électronique est réalisée pour prévoir son déplacement maximal. Grâce à ce travail, nous visons d'une part à étudier le déplacement maximal de la structure sous une charge thermique. D'autre part, à l'aide d'un couplage du calcul par éléments finis et un algorithme d'optimisation [**?**] au moyen d'un procédé hybride pour trouver la position optimale de la cinquième vis qui minimise le déplacement maximal de la carte électronique pour augmenter sa durée de vie espérance. Dans ce travail, nous avons montré que notre méthode donne de bons profits en utilisant le couplage d'un logiciel de calcul de chiffre et de la simulation et un outil de programmation. Nous avons l'intention d'examiner d'autres simulations en plaçant des composants sur la plaque et tester si le déplacement influe sur celle-ci

4.2.2 Optimisation du joint de brasure

Introduction

L'optimisation est devenue aujourd'hui un élément vital pour toutes les industries dans tous les domaines et l'utilisation d'algorithmes heuristiques a aidé à résoudre et surmonter de nombreux problèmes complexes qui ont été rencontrés. Dans cette étude, nous avons décidé d'utiliser l'hybridation de deux algorithmes : l'optimisation par essaim de particules et évolution différentielle afin d'optimiser la durée de la brasure d'une carte électronique [215]. Il est bien connu que les systèmes électroniques embarqués jouent un rôle majeur dans plusieurs domaines, tels que l'automobile, l'aérospatiale, les télécommunications, le médical ... etc. Pour assurer un fonctionnement correct et sûr, les systèmes électroniques doivent être fiables. Cependant, l'un des principaux problèmes des systèmes mécatroniques à bord est l'évaluation de leur fiabilité. Les causes de l'échec de ces systèmes sont multiples. Ils subissent de nombreuses contraintes de nature thermique extrême (variations de température), mécanique (chocs, vibrations) ... etc. Le mode de défaillance est observée en raison de cyclage thermique. En effet, lors du processus de production, les composants électroniques sont soumis à des températures allant de 250 ˚ C à -50 ˚ C. En outre, durant leur cycle de vie, les composants des systèmes embarqués sont soumis à des conditions de fonctionnement et des températures de l'environnement extérieur qui peuvent varier de -40 ˚ C à 150 ˚ C [70].

Ces cycles de température provoquent une déformation plastique dans les joints de soudure de composants de la carte électronique. Ces déformations plastiques peuvent provoquer l'apparition de fissures de fatigue et de joints de soudure des fractures. Cette perturbation entraîne une défaillance mécanique du composant de la carte électronique et une panne électrique des systèmes électroniques complets.

Cette étude vise principalement à prévoir la résistance à la fatigue des

joints de soudure en utilisant la simulation numérique par la méthode des éléments finis. L'optimisation est utilisée dans cette étude pour trouver une géométrie optimale de joint de soudure qui maximise la durée de vie de la fatigue. Dans ce travail, nous présentons une nouvelle méthode hybride d'optimisation par les algorithmes heuristiques pour évaluer la fiabilité de la carte électronique en simulant le comportement thermo-mécanique. Un modèle de simulation par éléments finis est développé pour étudier les déformations maximales en raison de la température, un couplage mécano-informatique est utilisé pour trouver la structure optimale. Pour effectuer correctement leurs fonctions, les systèmes électroniques doivent être fiables. Cet algorithme puissant et robuste qui repose sur l'hybridation de l'algorithme évolutionnaire différentiel avec l'optimisation par essaim de particules PSO donne des résultats intéressants [132].

Modèle de simulation de la carte électronique

La carte électronique est composée d'un type de micro-contrôleur 256 broches PLCC (Plastic Quad Flatpack) placé sur un circuit imprimé. Les joints sont en aile de mouette soudé sur le PCB (Printed Circuit Board) par une soudure sans plomb nommé SAC305 (96,5% d'étain, 3% d'argent et 0,5% de cuivre). Le modèle 2D par les éléments finis de la zone critique est développé, ce joint de soudure le plus sollicité par la charge thermique. Le modèle géométrique (figure 4.7) se compose d'une plaque de circuit imprimé (PCB) en FR4, une composante dans une résine EPOXY, une broche en cuivre et une soudure dans SAC305.
La figure 4.8 montre le maillage utilisé pour la simulation thermoméca-

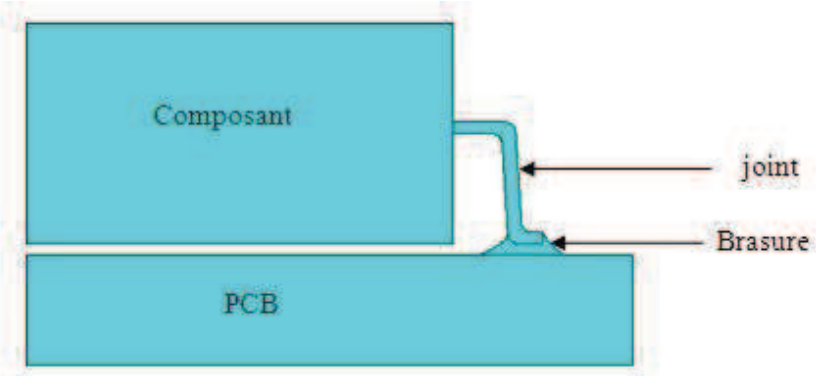

FIGURE 4.7 – Modèle géometrique

nique de la brasure de la carte électronique. notre intérêt se concentre sur le joint de soudure, le maillage est raffiné dans cette zone particulière.
Les propriétés des matériaux constituant la structure étudiée sont présentées dans le tableau 4.3.

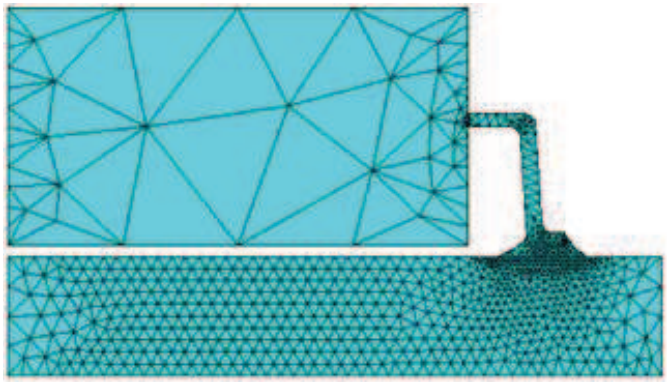

FIGURE 4.8 – Mallaige

Propriètés des matériaux	SAC305	FR4	resin EPOXY	copper
Modules de Young (GPa)	51.3	17	17	115
Coefficient de Poisson	0.35	0.39	0.24	0.31
Densité (Kg/m3)	7400	1800	1800	8890
Coefficient de dilatation thermique Expansion (mm/K)	20	18	22	17
Module de cisaillement(GPa)	19	2.4	7.4	44

TABLE 4.3 – Propriétés des matériaux

Le cycle de température appliqué à la structure est d'une durée de 60 minutes. La température varie entre -55 ° C et 125 ° C en fonction du profil présenté sur la figure 4.8.

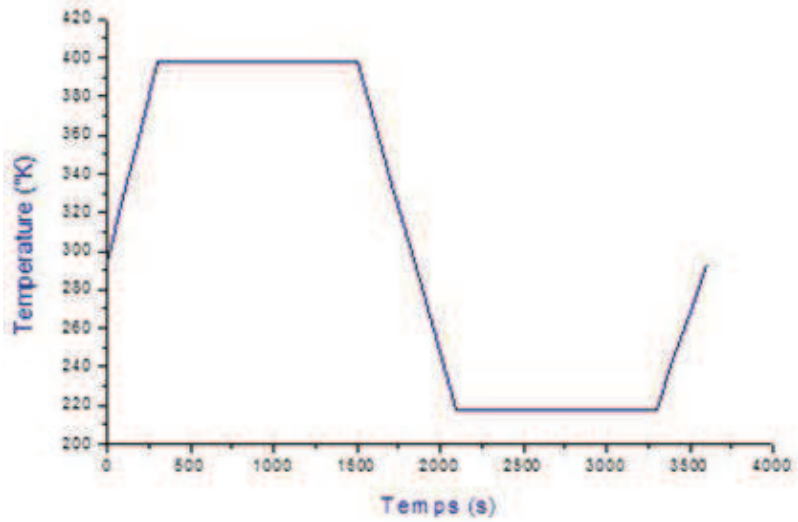

FIGURE 4.9 – Cycle de température

Modèle mathématique

Les variables d'optimisation sont les trois paramètres les plus influents dans les structures. Ces variables sont présentées dans 4.12 : épaisseur de la brasure E, L et H sont respectivement : la longueur et la hauteur de la broche. Le problème d'optimisation revient à maximiser la durée de vie en fatigue Nf :

$$\begin{cases} \text{Maximiser} f(x) = maxNf \\ \text{sous les contraintes} \\ 0.1 \leq E \leq 0.25 \\ 0.5 \leq L \leq 0.1 \\ 0.1 \leq H \leq 0.4 \end{cases} \tag{4.2}$$

Résultats et discussions

Dans cette étude, le nombre de cycles de vie en fatigue des joints à brasure tendre est estimé en fonction de la vitesse de déformation élastique. Dans un premier temps, nous présentons la distribution de la déformation élastique de l'articulation et dans un deuxième temps, nous déteminons le nombre de cycles de fatigue du joint de soudure. Enfin, nous allons faire une comparaison du nombre de cycles de fatigue du joint de soudure entre

la structure initiale et la structure optimale.

La déformation plastique

Le calcul de la déformation élastique de la brasure à la figure 4.10 indique les zones fortement déformées. L'épaisseur et la largeur de la soudure sont alors des dimensions caractéristiques puisque toutes les déformations inélastiques se produisent dans cette zone. L'identification de ces longueurs caractéristiques a été effectuée pour fournir le modèle d'optimisation.

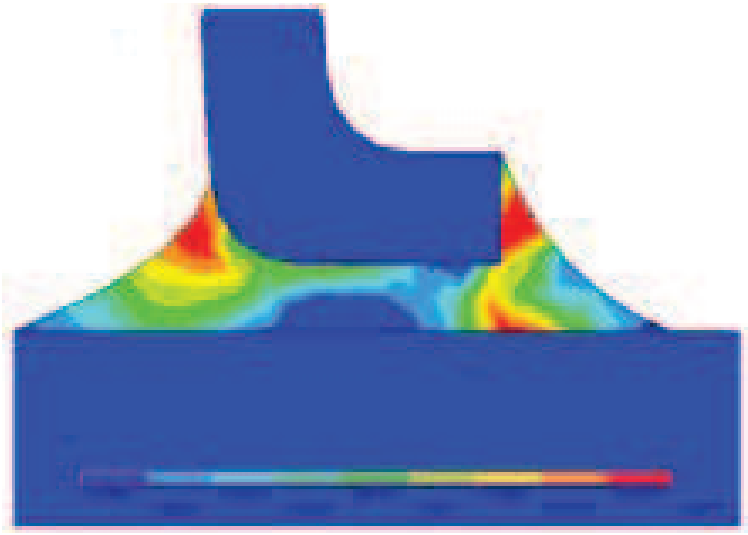

FIGURE 4.10 – Répartition de déformation inélastique dans la brasure

Couplage mécano-informatique

La méthodologie adoptée pour réaliser le couplage entre Matlab et Ansys pour trouver les paramètres géométriques optimaux du joint de soudure est représentée dans la figure 4.11.
Les variables d'optimisation sont les trois paramètres les plus influents dans les structures, la figure 4.12 : épaisseur E de la brasure, L et la longueur de la broche H et la hauteur de la broche.
Les résultats obtenus avec la méthode d'optimisation présentée ici sont comparés dans le tableau 4.4 avec les résultats déterministes.
Nous notons que la méthode d'optimisation heuristique donne de bons résultats meilleurs que la méthode déterministe et a permi à notre joint de

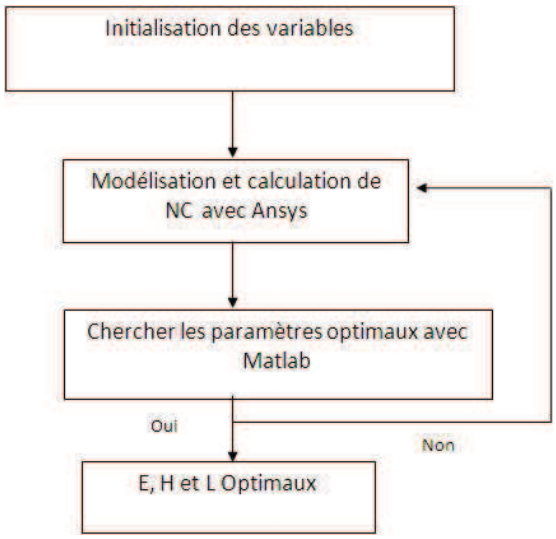

FIGURE 4.11 – Méthodologie d'optimisation

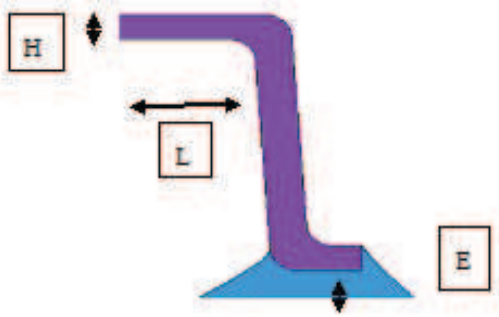

FIGURE 4.12 – Variables d'optimisation

Paramètres	Résultats déterministes	Résultats évolutionnaires
E (mm)	0.15	0.17
L (mm)	0.7	0.75
H (mm)	0.17	0.17
Nf	676.45	899

TABLE 4.4 – Comparaison des résultats déterministes et évolutionnaires

brasure un bénéfice commun de 222.55 cycles de vie, cet avantage donne plus longue vie de la carte électronique.

Concluion

Ce travail vise à étudier la fatigue des composants d'un joint de brasure SAC305 de type électroniques soumis à des contraintes sévères complexes de type de cyclage thermique. Un modèle basé sur les éléments finis est développé pour simuler le comportement viscoplastique d'alliage SAC305 et de déterminer la déformation élastique dans les domaines critiques de la brasure. Les résultats sont utilisés pour prédire la durée de vie de la fatigue. Nous avons montré que notre méthode, basée sur l'optimisation qui permet de trouver les valeurs optimales de la structure. Ces résultats contribueront à améliorer la vie du joint de soudure et par conséquent la durée de vie de la carte électronique en tenant compte de l'ensemble électronique.

Chapitre 5

Applications en optimisation multiobjectif

5.1 Introduction

L'optimisation multiobjectif des structures mécanique est un domaine d'actualité de la recherche scientifique. Elle essaye de traiter des problème à plusiers fonctions objectif complexes et le résoudre en utilisant des méthodes simpliées mais robustes. Le champ d'application de l'optimisation structurale s'étend aujourd'hui à de nouveaux défis toujours plus intéressants. La détermination de formes adéquates des composants structuraux est un problème de première importance pour l'ingénieur. Dans tous les domaines de la mécanique des structures, l'impact de la bonne conception d'une pièce est très important sur sa résistance, sa durée de vie et son utilisation en service. Ce défi est quotidien dans les secteurs de pointe tels que la recherche spatiale, électronique, l'aéronautique, l'automobile, la construction navale, la mécanique de précision et les ouvrages d'art en génie civil. Le développement de l'art de l'ingénieur requiert des efforts considérables pour améliorer sans cesse les techniques de conception des structures. L'optimisation intervient de façon primordiale dans l'augmentation des performances soit de masse, de position et aussi de coût.

Ce chapitre est consacré pour soulever les problèmes multi-objectif dans les domaines de la mécatronique et méchanotronique. Il traite deux types d'application, une application méchano-électronique (carte électronique) et l'autre mécanique sur les pylônes haubanés.
Ces travaux ont été menés dans le cadre de la collaboration avec les deux laboratoires LERMA de l'EMI et LOFIMS de l'INSA.

5.2 Optimisation méchano-électronique

Dans cette application le dispositif étudié est le boitier électronique (type de liaison par fil). Ce dispositif a la configuration de la pile. Les puces de silicium sont brasées dans les substrats généralement faites d'une couche de céramique (Al2O3 ou AIN) prise en sandwich entre deux couches minces de cuivre isolant, et la totalité de l'assemblage de la matrice et la servitude de cuivre directe (DCB) est soudée à une plaque de base en cuivre. les obligations de fil d'or et les barres de bus relient les terminaux, le substrat et le module. les dessins schématiques de la conception, du modèle sont donnés dans la figure 5.1.

Un des mécanismes de défaillance les plus élémentaires de dispositif de

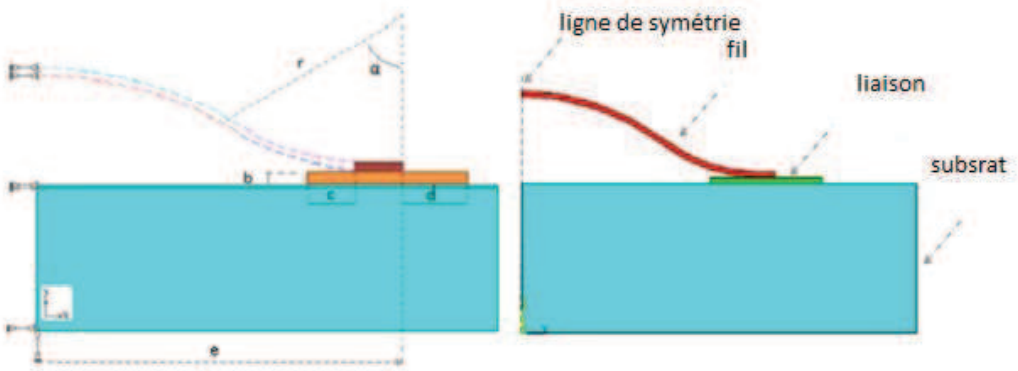

FIGURE 5.1 – Modèle simple de conception

puissance vient de fil de liaison décollé. La cause principale de ce défaut est les différences de coefficient de dilatation thermique entre le silicium et l'aluminium qui induit à un niveau élevé de contrainte thermo-mécanique à chaque fois que l'alimentation est l'objet d'une variation de température. Une telle contrainte thermomécanique répétée engendre une déformation plastique entre la métallisation de source et de fils de liaison et conduit à un décollement de liaison à l'extrémité du fil. La performance de ces structures dépend de la conception, l'objectif de cette application est de développer une méthodologie d'optimisation de conception pour améliorer la performance et la fiabilité du dispositif de puissance.

Le problème de l'optimisation de la conception de l'interconnexion de liaison par fil peut être écrit comme suit :

L'objectif est de minimiser $G(x) = [G_1(x), G_2(x), G_3(x)]$ avec le point initial : $x_i = [30^{\circ}, 250um, 250um, 1250um, 25um]$

Le concept est de trouver le motif qui maximise la fiabilité du système, afin de minimiser la contrainte de cisaillement à l'interface entre le plot de connexion et le substrat ($G_1(x)$), minimiser la contrainte de cisaillement à l'interface entre le fil et le plot de connexion ($G_2(x)$), et de réduire la

Propriétiés	Elastique Modul (GPa)	v	CTE (ppm/˚C)	Densité (gm/cm3)
Substrate Al2O3	300	0.21	7.1	3.9
Gold wires	E=97 (1-T/2032)	0.42	14.2	19.32

TABLE 5.1 – Propriétés des matériaux pour les couches adhérentes

contrainte de flexion du fil ($G_3(x)$).

$$\begin{cases} \text{Minimiser } G(x) = [G_1(x), G_2(x), G_3(x)] \\ \text{sous les contraintes} \\ x_1 = 30 \\ x_2 = 250 \\ x_3 = 250 \\ x_4 = 1250 \\ x_5 = 25 \end{cases} \tag{5.1}$$

Le logiciel de codage par éléments finis, ANSYS, a été utilisée pour calculer les trois fonctions objectifs. Pour éviter les calculs fastidieux et ressources informatiques expressives, une analyse en deux dimensions est réalisée dans cette étude. Les dimensions et les conditions aux limites du modèle numérique sont présentées dans la figure 5.1. Les conditions aux limites symétriques ont été appliquées le long du bord gauche du modèle. Le coin inférieur gauche a été contraint dans le sens vertical pour empêcher le mouvement du corps. L'ensemble est soumis à 25˚C à 500˚C de température. Le but de l'analyse FEM est de trouver la distribution de contrainte de déformation dans la soudure de fils sous chargement thermique cyclique. Les propriétés des matériaux utilisés dans ce modèle sont présentés dans le Tableau 5.1. Deux types de matériaux ont été utilisés pour décrire le comportement du matériau de la structure : la température des propriétés des matériaux dépendent de fil d'or obtenues à partir des courbes de contrainte-déformation dans [137].
Dans cette application, la nouvelle méthode hybride est utilisée pour résoudre ce problème d'optimisation de la conception, le tableau 5.2 montre la conception optimale et intial, ainsi, les résultats montrent que les trois fonctions objectifs (la contrainte de l'arbre) sont minimisés.

Les figure 5.2 et 5.3 montrent la nouvelle répartition de Von Mises Stress après l'application de la nouvelle méthode hybride de la procédure d'optimisation de la conception. Le stress est plus homogène que la distribution dans la configuration initiale.

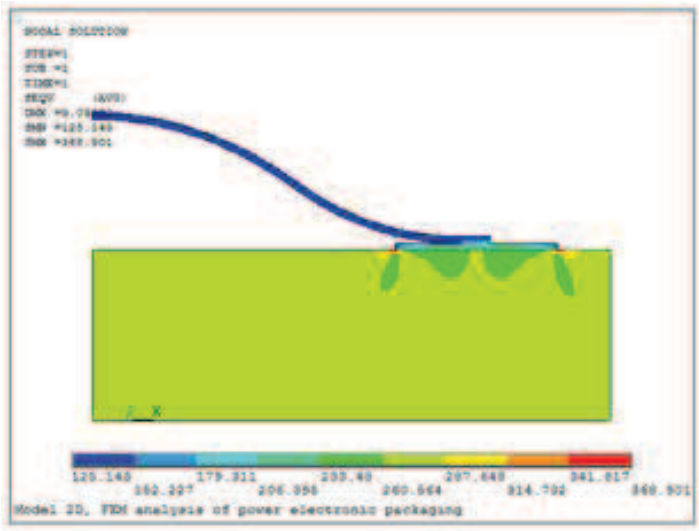

FIGURE 5.2 – Conception initiale

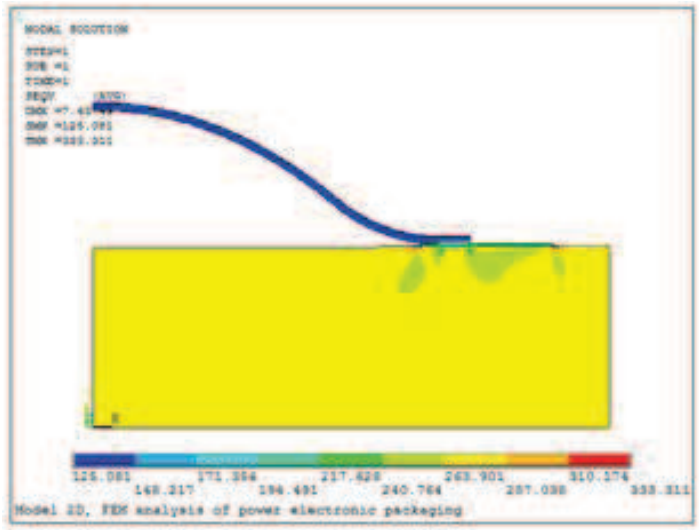

FIGURE 5.3 – Conception optimale

Description et unité	Point initial	Point optimal
b(μm)	25	11.5
c(μm)	250	75
d(μm)	250	294.84
α	30°	36°
r(μm)	1250	650.1
$G_1(x)$	169	129
$G_2(x)$	125	116
$G_3(x)$	142	125

TABLE 5.2 – Conception des résultats d'optimisation

5.3 Conclusion

Ce travail vise à étudier la maximisation de la fiabilité du système afin de minimiser la contrainte de cisaillement à l'interface entre plot de liaison et le substrat ($G_1(X)$), minimiser la contrainte de cisaillement à l'interface entre le fil et le plot de liaison ($G_2(X)$), et de minimiser la contrainte de flexion du fil ($G_3(X)$).

Un modèle par éléments finis est développé pour simuler le comportement viscoplastique de l'alliage commune et de déterminer la déformation élastique dans les domaines critiques de la brasure. Les résultats sont utilisés pour prédire la durée de vie de fatigue. Nous avons montré que notre méthode, basée sur l'optimisation de trouver les valeurs optimales de la structure. Ces résultats contribueront à améliorer la vie du joint de soudure. Par conséquent la durée de vie de la carte électronique en tenant compte de l'ensemble de la carte électronique.

5.4 Optimisation mécanique

L'étude dynamique non linéaire d'un pylône haubané suite à la rupture soudaine d'un câble de la nappe supérieure en présence de vent bissecteur nécessite un temps de calcul assez important [149]. La réalisation d'un calcul d'optimisation nécessitant un grand nombre de tests devient donc impossible. C'est dans cet objectif qu'il a été choisi de créer un modèle simplifié reproduisant le comportement dynamique du pylône haubané après rupture de câble : l'étude d'un pendule amorti.
les structures de conception, la réalisation et la sécurité de grandes hauteurs, comme un pylône haubanée, exigent une bonne connaissance de leur comportement à l'égard des forces extérieures au hasard (vent, séisme ...), y compris dans la plupart des cas extrêmes. La rupture d'un cable est l'un des plus grave problèmes et mérite d'être étudiée profondement.
Cette partie de ce chapitre est consacrée pour etudier l'optimisation mul-

tiobjectif d'un pylône haubané, l'objectif de ce travail est de minimiser le temps de calcul et aussi la minimisation d'amplitudes du pylône.

5.4.1 Pendule double

Configuration de l'étude

Le pendule double présenté dans la Figure 5.4 modélise le comportement d'un pylône haubané auquel un amortisseur a été introduit.

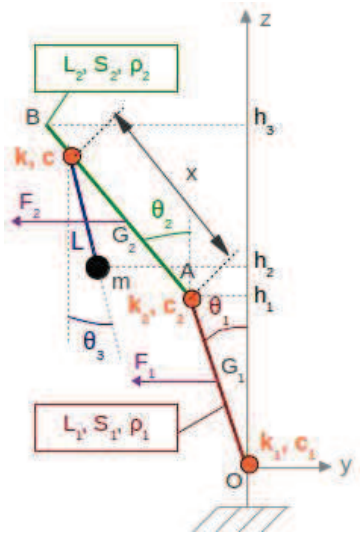

FIGURE 5.4 – Configuration du pendule double avec un amortisseur

5.4.2 Pendule triple

Configuration de l'étude

La configuration du pendule double ne représentant pas une grande sensibilité aux perturbations apportées aux paramètres de l'amortisseur, un second pendule inversé est étudié, constitué cette fois-ci de trois tiges. Les deux figures 5.5 et 5.6 illustrent les deux cas sur lesquels nous avons effectué notre optimisation qui réside dans la minimisation du temps de calcul et aussi la minimisation d'amplitudes du pylône haubané.
Le medèle mathématique est le suivant :

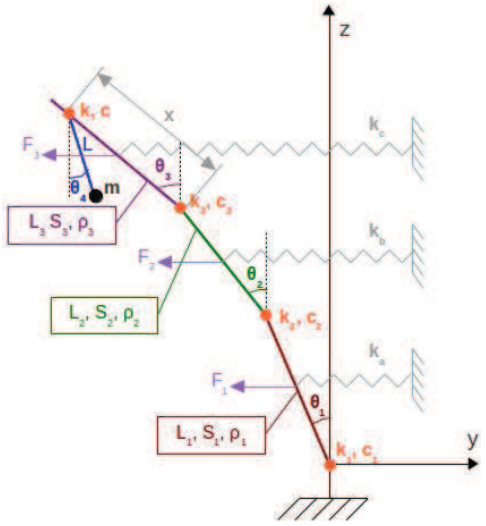

FIGURE 5.5 – Configuration à un amortisseur

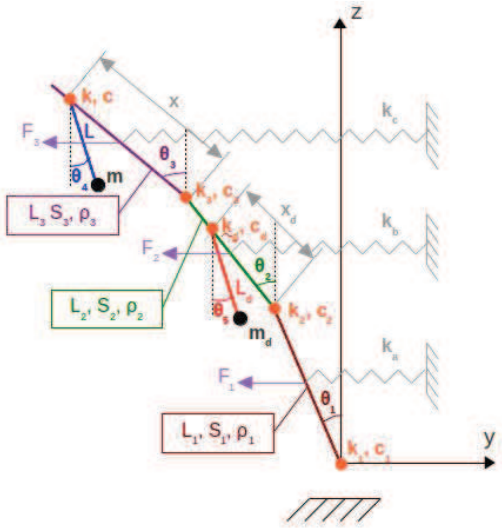

FIGURE 5.6 – Configuration à deux amortisseurs

$$\begin{cases} \text{Minimiser} A(X) = [A_1(X), A_2(X), A_3(X)] \\ \text{sous les contraintes} \\ x_1 = 1 \\ x_2 = 0.3 \\ x_3 = 0.0001 \\ x_4 = 0.2 \\ x_5 = 0.01 \end{cases} \tag{5.2}$$

Avec $A_1(x)$: la 1ère fonction objectif (l'amplitude du pendule 1).
$A_2(x)$: la 2éme fonction objectif (l'amplitude du pendule 2).
$A_3(x)$: la 3éme fonction objectif (l'amplitude du pendule 3).
Le tableau 5.3 résume les résultats obtenus :
Les ressorts et les ammortisseurs sont la pour rassurer la stabilité dU pen-

	Fmin	Time	nIter
Pendule à un amortisseur	0.5189	2.25322	150
pendule à double amortisseurs	0.494	3.569984	80

TABLE 5.3 – Comparaison des résultats dans le cas d'un seul et double amortisseurs

dule inversé ; le pendule est en équilibre stable grâce à la présence des ressorts et ammortisseurs. Les forces appliquées perturbent cet équilibre.

5.5 Conclusion

Depuis quelques années, il y a un intérêt croissant pour le contrôle actif en génie civil, permettant d'atténuer les effets des sollicitations dynamiques sur des structures telles que des buildings, des ponts, des plates-formes offshore, des tours de contrôle, ou des grues. Les sollicitations peuvent provenir du vent, du traffic, des tremblements de terre ou des vagues.
L'objectif de notre travail est la maximisation de la durée de vie des câbles de pylône par la minimisation des amplitudes d'oscillation. En premier lieu l'idée était l'introduction d'un amortisseur mais après plusieurs calculs nous avons constaté que l'insertion du double amortisseur sur le triple pylône donne des résultats méilleurs au niveau de la fonction coût même si le temps augmente faiblement.

Conclusion générale et perspectives

Les travaux de recherche présentés dans cette thèse concernent le développement de nouvelles méthodes d'optimisation globale qui s'appuient sur les algorithmes métaheuristiques. Nous avons focalisé nos recherches sur les algorithmes évolutionnaires dits stochastiques destinés à résoudre des problèmes d'optimisation difficiles.
Les applications traitées dans cette thèse entrent dans le cadre de l'optimisation en mécanique de structures et l'optimisation des structures mécatronique.
Notre première contribution est le couplage de l'algorithme génétique GA avec l'algorithme par Essaims particulaires (Particle Swarm Optimization) PSO. Le couplage de ces deux algorithmes est fait au niveau de l'opérateur de mutation dans GA. Cette technique hybride intègre des concepts de GA, de DE et des individus PSO et crée une nouvelle génération, non seulement par les opérateurs de croisement et mutation que l'on trouve dans les GA, mais aussi par des mécanismes de PSO afin d'obtenir une meilleure fonction objectif.
La deuxième contribution concerne le couplage d'algorithme d'évolution differentielle, ceci est un couplage des opérateurs de DE en hybridation par insertion de PSO, cette méthode a un avantage par rapport à la première contribution. Cet avantage réside au niveau d'une énorme minimisation de temps de calcul.
A partir des deux premières contributions nous avons abouti à rendre l'application des algorithmes évolutionnaires sur des problèmes multiobjectif plus facile. En effet, nous avons couplé les méthodes NBI ou NNCM avec nos nouveaux algorithmes pour sortir du cadre mono objectif et affronter les problèmes multiobjectifs.
Cette thèse traite quatre applications en mécanique et mécatronique ; la première application est l'optimisation de la position de la vis d'une carte électronique, ce problème est mono-objectif, son but est de trouver le bon positionnement de la vis de fixation de la carte électronique. Sachant que

ce positionnement résulte de la minimisation du déplacement maximale qui déforme notre carte.
La deuxième application entre toujours dans le cadre de la mécatronique, elle étudie le problème de l'optimisation de joint de brasure. Ce travail consiste à maximiser la durée de vie de la joint de brasure par conséquent la minimisation du coût de production.
La troisième application est la minimisation de trois fonctions objectifs, et vise à maximiser de la fiabilité du système afin de minimiser la contrainte de cisaillement à l'interface entre plot de liaison et le substrat, minimiser la contrainte de cisaillement à l'interface entre le fil et le plot de liaison , et minimiser la contrainte de flexion du fil .
La quatrième application est la maximisation de la durée de vie des câbles de pylône par la minimisation des amplitudes d'oscillation en premier lieu l'idée était l'introduction d'un amortisseur mais après plusieurs calcul nous avons constaté que l'insertion du double amortisseur sur le triple pylône nous donne des bons résultats au niveau de la fonction coût même si le temps augmente faiblement.

Perspectives

Certes les méthodes présentées dans cette thèse ont donné des résultats remarquables et pertinent mais la recherche continue !
Nous proposons d'hybrider dans le futur d'autres algorithmes heuristiques notemment les algorithmes harmoniques et de musique.
Nous comptons travailler sur des problèmes multiobjectifs plus complexes pour bien amliorer les limites de nos méthodes.
Dévlopper des métaheuristiques dans le cadre d'optimisation avec incertitude. Etablir un nouveau algorithme qui se base sur le mécanisme féminin.
S'ouvrir sur d'autres domaines comme la sécirité informatique et la bio informatique.

"Choisis un travail que tu aimes, et tu n'auras pas à travailler un seul jour de ta vie" Confucius

Bibliographie

[1] Analysis of the sequences within and flanking the cyanoglobin-encoding gene, glbn, of the cyanobacterium nostoc commune. *Gene*, 146(1) :133 – 134, 1994.

[2] Analyzing synchronous and asynchronous parallel distributed genetic algorithms. *Future Generation Computer Systems*, 17(4) :451 – 465, 2001.

[3] Solving an integrated employee timetabling and job-shop scheduling problem via hybrid branch-and-bound. *Computers & Operations Research*, 36(8) :2330 – 2340, 2009.

[4] Ant colony input parameters optimization for multiuser detection in ds/cdma systems. *Expert Systems with Applications*, 39(17) :12876 – 12884, 2012.

[5] B. Ait Brik. Méthodologies de conception robuste et d'optimisation dans un contexte de conception d'architectures mécaniques nouvelles en avant projet. *Thèse de Doctorat de l'Université de Franche-Comté*, 2005.

[6] Enrique Alba, El ghazali Talbi, and Albert Y. Zomaya. Nature-inspired distributed computing. *Computer Communications*, 30(4) :653 – 655, 2007.

[7] W. El Alem, A. El Hami, and R. Ellaia. A global stochastic optimization method for large scale problems. *Math. Model. Nat. Phenom.*, 5 :97–102, 2010.

[8] W. El Alem, A. El Hami, and R. Ellaia. A new methodology for an optimal shape design. *International Journal Applied Mechanics and Materials*, to apper, 2011.

[9] G. Allaire. Conception optimale de structures, mathématiques et applications. *Springer*, 58, 2007.

[10] ANSYS. Ansys structural analysis guide. *ANSYS Release 10.0. ANSYS, Inc., Canonsburg, Pensilvania, USA*, 2005.

[11] David L. Applegate, William Cook, Sanjeeb Dash, and Daniel G. Espinoza. Exact solutions to linear programming problems. *Operations Research Letters*, 35(6) :693 – 699, 2007.

[12] J.S. Arora. Introduction to optimum design. *McGraw-Hill, New York.*, 1989.

[13] J.S. Arora, E.J. Haug. Methods of design sensitivity analysis in structural optimization. *AIAA Journal*, 19 :1761–1783, 1979.

[14] M. S. Arumugam, M. V. C. Rao, and A. W. C. Tan. A novel and effective particle swarm optimization like algorithm with extrapolation technique. *Applied Soft Computing*, 9 :308–320, 2009.

[15] C. Audet, J.E.Jr. Dennis. Analysis of generalized pattern searches. *SIAM Journal on Optimization*, 13 :889–903, 2003.

[16] C. Audet, J.E.Jr. Dennis. Mesh adaptive direct search algorithms for constrained optimization. *SIAM Journal on Optimization*, 17 :188–217, 2006.

[17] P. Augerat, J.M. Belenguer, E. Benavent, A. CorberÃ¡n, and D. Naddef. Separating capacity constraints in the {CVRP} using tabu search. *European Journal of Operational Research*, 106(2-3) :546 – 557, 1998.

[18] R. Azencott. *Simulated Annealing : Parallelization Techniques*. 1992.

[19] Zahra Naji Azimi. Hybrid heuristics for examination timetabling problem. *Applied Mathematics and Computation*, 163(2) :705 – 733, 2005.

[20] Philippe Badeau, FranÃ§ois Guertin, Michel Gendreau, Jean-Yves Potvin, and Eric Taillard. A parallel tabu search heuristic for the vehicle routing problem with time windows. *Transportation Research Part C : Emerging Technologies*, 5(2) :109 – 122, 1997. <ce :title>Parallel Computing in Transport Research</ce :title>.

[21] A. Belegundu, J. Arora. A recursive quadratic programming method with active set strategy for optimal design. *International Journal for Numerical Methods in Engineering*, 20 :803–816, 1984.

[22] A.D. Belegundu. A study of mathematical programming methods for structural optimization. *Department of Civil and Environmental Engineering, University of Iowa, Iowa City, Iowa.*, 1982.

[23] R. Bellman. Adaptive control processes : A guided tour. *Princeton University Press*, 1961.

[24] A. Bendjoudi, N. Melab, and E-G. Talbi. An adaptive hierarchical masterworker (ahmw) framework for grids application to bamp algorithms. *Journal of Parallel and Distributed Computing*, 72(2) :120 – 131, 2012.

[25] J.A. Bennet, M.E. Botkin. Shape optimization of two-dimensional structures with geometric problem description and adaptative mesh refinement. *AIAA Journal*, 23 :458–464, 1983.

[26] A. Berro. Optimisation multiobjectif et stratégies d'évolution en environnement dynamique. *Thèse de Doctorat de l'Université des Sciences Sociales Toulouse I*, 2001.

[27] J. R. Birge, F. Louveaux. Introduction to stochastic programming. *Springer-Verlag, New York*, 1997.

[28] Christian Blum, Jakob Puchinger, Ganther R. Raidl, and Andrea Roli. Hybrid metaheuristics in combinatorial optimization : A survey. *Applied Soft Computing*, 11(6) :4135 – 4151, 2011.

[29] P. T. Boggs, J. W. Tolle. Sequential quadratic programming for large-scale nonlinear optimization. *Journal of Computational and Applied Mathematics*, 124 :123–137, 2000.

[30] E. Bonabeau, M. Dorigo, and G. Theraulaz. Swarm intelligence : from natural to artificial systems. *Oxford University Press*, 1999.

[31] A.J. Booker, J.E.Jr. Dennis, P.D. Frank, D.B. Serafini, V. Torczon, and M.W. Trosset. A rigorous framework for optimization of expensive functions by surrogates. *Structural Optimization*, 17 :1–13, 1999.

[32] M.E. Botkin, J.A. Bennet. Shape optimization of three dimensional folded plate structures. *AIAA Journal*, 23 :1804–1810, 1985.

[33] Ines Bouhlel, Hedi Ben Mansour, Ilef Limem, Mohamed Ben Sghaier, Amor Mahmoud, Jemni Ben Chibani, Kamel Ghedira, and Leila Chekir-Ghedira. Screening of antimutagenicity via antioxidant activity in different extracts from the leaves of acacia salicina from the center of tunisia. *Environmental Toxicology and Pharmacology*, 23(1) :56 – 63, 2007.

[34] S. Boyd, L. Vandenberghe. Convex optimization. *Cambridge University Press*, 2004.

[35] V. Braibant, C. Fleury. Shape optimal design using b-splines. *Computer Methods in Applied Mechanics and Engineering*, 44 :247–267, 1984.

[36] C.G. Broyden. Quasi-newton methods and their applications to function minimization. *Mathematics of Computation*, 21 :368–381, 1967.

[37] C.G. Broyden. The convergence of a class of double rank minimization algorithms. *J.Inst. Maths Applics*, 6 :222–231, 1970.

[38] M. Bruyneel, P. Duysinx, and C. Fleury. A family of mma approximations for structural optimization. *Struct Multidisc Optim*, 24 :263–276, 2002.

[39] S. Calvel. Conception d'organes automobiles par optimisation topologique. *Thèse de Doctorat de l'Université Paul Sabatier toulouse III*, 2004.

[40] A. Canelas, J. Herskovits, and J.C.F. Telles. Shape optimization using the boundary element method and a sand interior point algorithm for constrained optimization. *Computers and Structures*, 86 :1517–1526, 2008.

[41] V. Cerny. Thermodynamical approach to the traveling saleman problem : an efficient simulation algorithm. *J. of Optimization th. and applications*, 45 :41–51, 1985.

[42] Heming Chan, P. Mazumder, and K. Shahookar. Macro-cell and module placement by genetic adaptive search with bitmap-represented chromosome. *Integration, the {VLSI} Journal*, 12(1) :49 – 77, 1991.

[43] A. Charnes, W.W. Cooper, and G.H. Symonds. Cost horizons and certainty equivalents : An approach to stochastic programming of heating oil. *Management Science*, 4 :183–195, 1958.

[44] A. Charnes, W. Cooper. Management models and lndustrial applications of linear programming. *John Wiley. NewYork*, 1 :425–460, 1961.

[45] R. Chelouah and P. Siarry. A hybrid method combining continuous tabu search and nelder-mead simplex algorithms for the global optimization of multiminima functions. *European Journal of Operational Research*, 161 :636–654, 2005.

[46] Y. L. Chen, C. C. Liu. Multiobjectif var planning using the goal attainment method. *Proceedings on Generation, Transmission and Distribution*, 141 :227–232, 1994.

[47] M. Clerc, J. Kennedy. The particle swarm : explosion, stability, and convergence in multidimensional complex space. *IEEE Trans. on Evolutionary Compt.*, 6 :58–73, 2002.

[48] C. A. C. Coello. An empirical study of evolutionary techniques for multiobjective optimization in engineering design. *Ph. D. thesis, Department of Computer Science, Tulane Universitv. New Orleans*, 1996.

[49] C.A.C. Coello. Use of a self-adaptive penalty approach for engineering optimization problems. *Computers in Industry*, 41 :113–127, 2000.

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

[51] G. Cohen. Convexité et optimisation. *Ecole Nationale des Ponts et Chaussées et INRIA, http ://cel.archives-ouvertes.fr/docs/00/35/66/86/PDF/Ponts-cours-A4-NB.pdf*, 2000.

[52] Lucio Comai, Naoko Tanese, and Robert Tjian. The tata-binding protein and associated factors are integral components of the {RNA} polymerase i transcription factor, {SL1}. *Cell*, 68(5) :965 – 976, 1992.

[53] A.R. Conn, K. Scheinberg, and Ph.L. Toint. Recent progress in unconstrained nonlinear optimization without derivatives. *Mathematical Programming*, 79 :397–414, 1997.

[54] A.R. Conn, K. Scheinberg, and L.N. Vicente. Geometry of interpolation sets in derivative free optimization. *Mathematical Programming*, 111 :141–172, 2008.

[55] Carlos Cotta. Scatter search with path relinking for phylogenetic inference. *European Journal of Operational Research*, 169(2) :520 – 532, 2006. <ce :title>Feature Cluster on Scatter Search Methods for Optimization</ce :title>.

[56] A.L. Custódio, J.E.Jr. Dennis, and L.N. Vicente. Using simplex gradients of nonsmooth functions in direct search methods. *IMA Journal of Numerical Analysis*, 28 :770–784, 2008.

[57] A.L. Custódio, L.N. Vicente. Using sampling and simplex derivatives in pattern search methods. *SIAM Journal on Optimization*, 18 :537–555, 2007.

[58] Nadia Dahmani, François Clautiaux, Saoussen Krichen, and El-Ghazali Talbi. Iterative approaches for solving a multi-objective 2-dimensional vector packing problem. *Computers & Industrial Engineering*, (103) :–, 2013.

[59] Emilie Danna and David L. Woodruff. How to select a small set of diverse solutions to mixed integer programming problems. *Operations Research Letters*, 37(4) :255 – 260, 2009.

[60] G. B. Dantzig. Programmation linéaire et extensions. *Presses de l'Université Princeton, Prince-tonne, New Jersey*, 1963.

[61] G.B. Dantzig. Linear programming under uncertainty. *Management Science*, 1 :197–206, 1955.

[62] I. Das, J. E. Dennis. Normal-boundary intersection : a new method for generating the pareto surface in nonlinear multi-criteria optimization problems. *SIAM J. Optim.*, 8 :631–657, 1998.

[63] W. C. Davidon. Variable metric methods for minimization. *A.E.C. Research and Development Report ANL-5990 (Rev. TID-4500, 14th ed.)*, 1959.

[64] W.C. Davidon. Variable metric method for minimization. *SIAM Journal on Optimization*, 1 :1–17, 1991.

[65] K.A. De Jong. An analysis of behavior of a class of genetic adaptive systems. *ph.D. dissertation, Dept. Comput. Sci., The Univ. Michigan, Ann Arbor, MI*, 1975.

[66] K. Deb. Multiobjective genetic algorithms : problem difficulties and construction of test problems. *Evol. Comput.*, 7 :205–230, 1999.

[67] K. Deb, S. Agrawal, A. Pratab, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization : Nsga-ii. *in Proc. Conf. Parallel Problem Solving From Nature VI*, pages 849–858, 2000.

[68] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm : Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6 :182–197, 2002.

[69] K. Deep, M. Thakur. A new mutation operator for real coded genetic algorithms. *Applied Mathematics and Computation*, 193 :211–230, 2007.

[70] Shizuhiko Deji and Kunihide Nishizawa. Effects of high-frequency electromagnetic fields emitted from card readers of access control systems on electronic pocket dosimeters. *Applied Radiation and Isotopes*, 62(6) :951 – 953, 2005.

[71] C. Dhaenens, J. Lemesre, and E.G. Talbi. K-ppm : A new exact method to solve multi-objective combinatorial optimization problems. *European Journal of Operational Research*, 200(1) :45 – 53, 2010.

[72] L.C.W. Dixon, G.P. Szego. Towards global optimization 2. *North-Holland, Amsterdam*, 1978.

[73] M. Dorigo. Optimization, learning and natural algorithms. *Ph.D. Thesis, Politecnico di Milano, Italy*, 1992.

[74] M. Dorigo, V. Maniezzo, and A. Colorni. Ant system : optimization by a colony of cooperating agents. *IEEE Trans. on Man. Cyber. Part B*, 26 :29–41, 1996.

[75] M. Dorigo, C. Blum. Ant colony optimization theory : A survey. *Theoretical Computer Sc*, 344 :243–278, 2005.

[76] J. Dréo, A. Pétrowski, P. Siarry, and E. Taillard. Métaheuristiques pour l'optimisation difficile. *Eyrolles. Paris*, 2003.

[77] Ioana Dumitrescu, Patrick R. Unwin, and Julie V. Macpherson. Electrochemical impedance spectroscopy at single-walled carbon nanotube network ultramicroelectrodes. *Electrochemistry Communications*, 11(11) :2081 – 2084, 2009.

[78] R. Eberhart, J. Kennedy, and Y. Shi. Swarm intelligence. *Evolutionary Computation. Morgan Kaufmann*, 2001.

[79] R.C. Eberhart, J. Kennedy. A new optimizer using particle swarm theory. *Sixth International Symposium on Micro Machine and Human Science, Nagoya, Japan*, 1995.

[80] M. Ehrgott, X. Gandibleux. A survey and annotated bibliography of multiobjective combinatorial optimization. *OR Spektrum*, 22 :425–460, 2000.

[81] Z. El Maskaoui. Application de la programmation orientée objets à l'optimisation discrète sous contraintes des structures métalliques formées de poutres via les algorithmes génétiques. *Thèse de doctorat. Faculté polythechnique de mons*, 2007.

[82] S. Elaoud, T. Loukil, and J. Teghem. The pareto fitness genetic algorithm : Test function study. *European Journal of Operational Research*, 177 :1703–1719, 2007.

[83] Bouazza Elbenani, Jacques A. Ferland, and Jonathan Bellemare. Genetic algorithm and large neighbourhood search to solve the cell formation problem. *Expert Systems with Applications*, 39(3) :2408 – 2414, 2012.

[84] JosÃ© Rui Figueira and El-Ghazali Talbi. Emergent nature inspired algorithms for multi-objective optimization. *Computers & Operations Research*, 40(6) :1521 – 1523, 2013.

[85] J.R. Figueira, A. Liefooghe, E.-G. Talbi, and A.P. Wierzbicki. A parallel multiple reference point approach for multi-objective optimization. *European Journal of Operational Research*, 205(2) :390 – 400, 2010.

[86] R. Fletcher. A new approach to variable-metric algorithms. *Computer Journal*, 13 :317–322, 1970.

[87] R. Fletcher. Practical methods of optimization. *John Wiley & Sons, second edn*, 2000.

[88] R. Fletcher, M.J.D. Powell. A rapidly convergent descent method for minimization. *The Computer Journal*, 6 :163–168, 1963.

[89] C. Fleury. Méthodes numériques d'optimisation des structures. *rapport interne LTAS (Université de Liège) SF-19*, 1973.

[90] C. Fleury. Structural weight optimization by dual methods of convex programming. *International Journal for Numerical Methods in Engineerin*, 14 :1761–1783, 1979.

[91] C. Fleury. Efficient approximation concepts using second order information. *International Journal for Numerical Methods in Engineering*, 28 :2041–2058, 1989.

[92] C. Fleury, V. Braibant. Structural optimization : A new dual method using mixed variables. *International Journal for Numerical Methods in Engineering*, 23 :409–428, 1986.

[93] C. M. Fonseca and P. J. Fleming. Genetic algorithm for multiobjective optimization formulation, discussion and generalization. *In Proceedings of the Fifth International Conference on Genetic Algorithms, San Mateo. California*, pages 416–423, 1993.

[94] C. M. Fonseca, P. J. Fleming. Multiobjective optimization and multiple constraint handling with evolutionary algorithms-part i : a unified formulation. *IEEE Trans. System, Man, Cybern.*, 28 :26–37, 1998.

[95] Fourman. Compaction of symbolic layout using genetic algorithms. *In Genetic Algorithms and their Applications : Proceedings of the First International Conference on Genetic Algorithm*, pages 141–153, 1985.

[96] R.L. Fox. Constraint surface normals for structural synthesis techniques. *AIAA Journal*, 3 :1517–1518, 1965.

[97] M.A. French and G. Pritchard. The fracture surfaces of hybrid fibre composites. *Composites Science and Technology*, 47(3) :217 – 223, 1993.

[98] K.V. Frish. Vie et moeurs des abeilles. *Albin Michel*, 1984.

[99] J. Gablonsky, C.T. Kelley. A locally-biased form of the direct algorithm. *Journal of Global Optimization*, 21 :27–37, 2001.

[100] L.M. Gambardella, R. Montemanni, and D. Weyland. Coupling ant colony systems with strong local searches. *European Journal of Operational Research*, 220(3) :831 – 843, 2012.

[101] J. Garca-Nieto, E. Alba, L. Jourdan, and E. Talbi. Sensitivity and specificity based multiobjective approach for feature selection : Application to cancer diagnosis. *Information Processing Letters*, 109(16) :887 – 896, 2009.

[102] A. Georgieva and I. Jordanov. Global optimization based on novel heuristics, low-discrepancy sequences and genetic algorithms. *European Journal of Operational Research*, 196 :413–422, 2009.

[103] F. Glover. Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, 13 :533–549, 1986.

[104] F. Glover and M. Laguna. *Tabu Search*. 1997.

[105] T. Gmur. Méthode des éléments finis en mécanique des structures. *Presses polytechniques est universitaires normandes, Lausanne, Switzerland*, 2000.

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

[107] D. Goldfarb. A family of variable-metric methods derived by variational means. *Mathematics of Computation*, 24 :23–26, 1970.

[108] Stefan Gollowitzer and Ivana Ljubic. {MIP} models for connected facility location : A theoretical and computational study. *Computers & Operations Research*, 38(2) :435 – 449, 2011.

[109] W. Gong, Z. Cai, and L. Jiang. Enhancing the performance of differential evolution using orthogonal design method. *Applied Mathematics and Computation*, 206 :56–69, 2008.

[110] Dirk Van Gucht and Patrick C. Fischer. Multilevel nested relational structures. *Journal of Computer and System Sciences*, 36(1) :77 – 105, 1988.

[111] C. Hamzacebi and F. Kutay. A heuristic approach for finding the global minimum : Adaptive random search technique. *Applied Mathematics and Computation*, 173 :1323–1333, 2006.

[112] Q. He, L. Wang. An effective co-evolutionary particle swarm optimization for constrained engineering design problems. *Engineering Applications of Artificial Intelligence*, 20 :89–99, 2007.

[113] J. Herskovits, G. Dias, G. Santos, and C.M. Mota Soares. Shape structural optimization with an interior point nonlinear programming algorithm. *Struct Multidisc Optim*, 20 :107–115, 2000.

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

[115] R. Hooke, T.A. Jeeves. Direct search solution of numerical and statistical problems. *J. Assoc. Comput. Mach.*, 8 :212–229, 1961.

[116] J. Horn, N. Nafpliotis. Multiobjective optimisation using the niched-pareto genetic algorithm. *Illigal TR. n° 93005. Julv*, 1993.

[117] R. Horst, P. M. Pardalos, and H. E. Romeijn. Handbook of global optimization. *Springer Publisher, Kluwer*, 1995.

[118] Z. Huang, X. Miao, and P. Wang. A revised cut-peak function method for box constrained continuous global optimization. *Applied Mathematics and Computation*, 194 :224–233, 2007.

[119] J. Jie, J. Zeng, C. Han, and Q. Wang. Knowledge-based cooperative particle swarm optimization. *Applied Mathematics and Computation*, 205 :861–873, 2008.

[120] D.R. Jones, C.D. Perttunen, and B.E. Stuckman. Lipschitzian optimization without the lipschitz constant. *Journal of Optimization Theory and Application*, 79 :157–181, 1993.

[121] K.A. De. Jong. An analysis of the behaviour of a class of genetic adaptive systems. *PhD thesis, University of Michigan*, 1975.

[122] L. Jourdan, M. Basseur, and E.-G. Talbi. Hybridizing exact methods and metaheuristics : A taxonomy. *European Journal of Operational Research*, 199(3) :620 – 629, 2009.

[123] L. Jourdan, C. Dhaenens, E.-G. Talbi, and S. Gallina. A data mining approach to discover genetic and environmental factors involved in multifactorial diseases. *Knowledge-Based Systems*, 15(4) :235 – 242, 2002.

[124] Laetitia Jourdan, Clarisse Dhaenens-Flipo, and El-Ghazali Talbi. Chapter 14 - discovery of genetic and environmental interactions in disease data using evolutionary computation. In Gary B. Fogel and David W. Corne, editors, *Evolutionary Computation in Bioinformatics*, pages 297 – 316. Morgan Kaufmann, San Francisco, 2003.

[125] Nicolas Jozefowiez, Frédéric Semet, and El-Ghazali Talbi. The bi-objective covering tour problem. *Computers & Operations Research*, 34(7) :1929 – 1942, 2007.

[126] Nicolas Jozefowiez, Frédéric Semet, and El-Ghazali Talbi. Multi-objective vehicle routing problems. *European Journal of Operational Research*, 189(2) :293 – 309, 2008.

[127] Nicolas Jozefowiez, Frédéric Semet, and El-Ghazali Talbi. An evolutionary algorithm for the vehicle routing problem with route balancing. *European Journal of Operational Research*, 195(3) :761 – 769, 2009.

[128] Alan R. McKendall Jr. and Jin Shang. Hybrid ant systems for the dynamic facility layout problem. *Computers & Operations Research*, 33(3) :790 – 803, 2006.

[129] P. Kall, S.W. Wallace. Stochastic programming. *Wiley, Chichester etc.*, 1994.

[130] Y. Kao, E. Zahara. A hybrid genetic algorithm and particle swarm optimization for multimodal functions. *Applied Soft Computing*, 8 :849–857, 2008.

[131] S. Karmakara, S.K. Mahatob, and A.K. Bhunia. Interval oriented multi-section techniques for global optimization. *Journal of Computational and Applied Mathematics*, 224 :476–491, 2009.

[132] J. Kennedy. The behavior of particles. In *Proceedings of the 7th Conference on Evolutionary Computation, LNCS, Springer*, 1998.

[133] J. Kennedy, R.C. Eberhart. Particle swarm optimization. *IEEE International Conference on Neural Networks, Perth, Australia*, 1995.

[134] Ali Khanafer, François Clautiaux, Saad Hanafi, and El-Ghazali Talbi. The min-conflict packing problem. *Computers & Operations Research*, 39(9) :2122 – 2132, 2012.

[135] Ali Khanafer, François Clautiaux, and El-Ghazali Talbi. Tree-decomposition based heuristics for the two-dimensional bin packing problem with conflicts. *Computers & Operations Research*, 39(1) :54 – 63, 2012. <ce :title>Special Issue on Knapsack Problems and Applications</ce :title>.

[136] G. Kharmanda. *Optimisation et CAO des Structures Fiables*. PhD thesis, Université Blaise Pascal Clermont II, France, 2003.

[137] G. Kharmanda, A. Mohamed, and M. Lemaire. Carod : Computer aided reliable and optimal design as a concurrent system for real structures. *International Journal of CAD and CAM*, 2 :1–12, 2002.

[138] Mostepha R. Khouadjia, Briseida Sarasola, Enrique Alba, Laetitia Jourdan, and El-Ghazali Talbi. A comparative study between dynamic adapted {PSO} and {VNS} for the vehicle routing problem with dynamic requests. *Applied Soft Computing*, 12(4) :1426 – 1439, 2012.

[139] S.H. Kim and Y.K. Wen. Optimization of structures under stochastic loads. *Structural Safety*, 7 :177–190, 1990.

[140] S. Kirkpatrick, C. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220 :671–680, 1983.

[141] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by simulated annealing. *Science*, 220 :671–680, 1983.

[142] A. Der Kiureghian and J-B. Ke. The stochastic finite element method in structural reliability. *Prob. Engng. Mech.*, 3 :83–91, 1988.

[143] J. D. Knowles, D. W. Corne. The pareto archived evolution strategy : A new base une algorithm for multiobjective optimisation. *Congress on Evolutionary Computation, Washington. Julv*, pages 98–105, 1999.

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

[145] G. Kolomvos. Résolution de grands problèmes stochastiques multiétapes : Application à un problème de dimensionnement de capacités et de gestion de flux et de stocks. *ÉCOLE CENTRALE DES ARTS ET MANUFACTURES « ÉCOLE CENTRALE PARIS »*, 2007.

[146] James M. Krueger. Somnogenic activity of immune response modifiers. *Trends in Pharmacological Sciences*, 11(3) :122 – 126, 1990.

[147] DL. Kunz and AS. Hopkins. Structured data in structural analysis software. *Comput. Struct.*, 26 :965–978, 1987.

[148] I.E. Lagaris, I. G. Tsoulos. Stopping rules for box-constrained stochastic global optimization. *Applied Mathematics and Computation*, 197 :622–632, 2008.

[149] G.L Larose, A Zasso, S Melelli, and D Casanova. Field measurements of the wind-induced response of a 254 m high free-standing bridge pylon. *Journal of Wind Engineering and Industrial Aerodynamics*, 74-76(0) :891 – 902, 1998.

[150] K. S. Lee, Z. W. Geem. A new meta-heuristic algorithm for continuous engineering optimization : harmony search theory and practice. *Comput. Methods Appl. Mech. Engrg.*, 194 :3902–3933, 2005.

[151] M. Lemaire. *Fiabilité des structures : Couplage mécano-fiabiliste statique*. Hermes Science, 2005.

[152] J. Lemesre, C. Dhaenens, and E.G. Talbi. An exact parallel method for a bi-objective permutation flowshop problem. *European Journal of Operational Research*, 177(3) :1641 – 1655, 2007.

[153] L. Liberti, N. Maculan. Global optimization : from theory to implementation. *Springer Publisher*, 2006.

[154] A. Liefooghe, L. Jourdan, and E.-G. Talbi. Metaheuristics and cooperative approaches for the bi-objective ring star problem. *Computers & Operations Research*, 37(6) :1033 – 1044, 2010.

[155] Arnaud Liefooghe, Matthieu Basseur, Jérémie Humeau, Laetitia Jourdan, and El-Ghazali Talbi. On optimizing a bi-objective flowshop scheduling problem in an uncertain environment. *Computers & Mathematics with Applications*, 64(12) :3747 – 3762, 2012.

[156] H. Lin and K. Yamashita. Hybrid simplex genetic algorithm for blind equalization using {RBF} networks. *Mathematics and Computers in Simulation*, 59(4) :293 – 304, 2002.

[157] C. X. Liu and G. Choi. Structural optimization using matlab. *Engineering Systems and Design Lab. KAIST*, 2006.

[158] L. Liu, G. Wang. Designing superresolution optical pupil filter with constrained global optimization algorithm. *Optik*, 119 :481–484, 2008.

[159] H. Lu, G.G. Yen. Rank-density-based multiobjective genetic algorithm and benchmark test function study. *IEEE Transactions on Evolutionary Computation*, 7 :325–343, 2003.

[160] D. G. Luenberger. Introduction to linear and nonlinear programming. *Addison Wesley*, 1973.

[161] H. van der Vlerk Maarten. Stochastic programming bibliography. *World Wide Web, `http://www.eco.rug.nl/mally/spbib.html`, 1996-2007*.

[162] R. Martin. Single-interval learning by simile within a simulated hebbian neural network. *Computers & Mathematics with Applications*, 20(4-6) :217 – 226, 1990.

[163] J. Mathias, X. Balandraud, and M. Grediac. Applying a genetic algorithm to the optimization of composite patches. *Computers and Structures*, 84 :823–834, 2006.

[164] N. Melab, S. Cahon, and E-G. Talbi. Grid computing for parallel bioinspired algorithms. *Journal of Parallel and Distributed Computing*, 66(8) :1052 – 1061, 2006.

[165] N. Melab, M. Mezmaz, and E.-G. Talbi. Parallel cooperative meta-heuristics on the computational grid. : A case study : the bi-objective flow-shop problem. *Parallel Computing*, 32(9) :643 – 659, 2006. <ce :title>Optimization on Grids - Optimization for Grids</ce :title>.

[166] F. Messine. Méthodes d'optimisation globale basées sur l'analyse d'intervalle pour la résolution de problèmes avec contraintes. *Thèse de doctorat, Institut National Polytechnique de Toulouse*, 1997.

[167] N. Metropolis, M. Rosenbluth, A. Teller, and E. Teller. Optimization by simulated annealing, equation of state calculation by fast computing machines. *J. of Chemical Physic*, 21 :1087–1092, 1953.

[168] Z. Michalewicz, D. Dasgupta, R. Leriche, and M. Schoenauer. Evolutionary algorithms for constrained engineering problems. *Computers and Industrial Engineering Journal*, 30, 1996.

[169] N. Mladenovic, M. Drazic, V. Kovacevic-Vujcic, and M. Cangalovic. General variable neighborhood search for the continuous optimization. *European Journal of Operational Research*, 191 :753–770, 2008.

[170] R. Moore. Interval analysis. *Prentice Hall, NJ*, 1966.

[171] S. Mouelhi. Vers une chaîne cohérente de modèles pluie-débit conceptuels globaux aux pas de temps pluriannuel, annuel, mensuel et journalier. *Université Paris VI, ENGREF, Ecole des Mines de Paris*, 2003.

[172] J.A. Nelder, R. Mead. A simplex method for function minimization. *The Computer Journal*, 7 :308–313, 1965.

[173] J. Nocedal, S. J. Wright. Numerical optimization. *Springer Series in Operations Research, Springer*, 1999.

[174] D. Ouelhadj, S. Petrovic, P.I. Cowling, and A. Meisels. Inter-agent cooperation and communication for agent-based robust dynamic scheduling in steel production. *Advanced Engineering Informatics*, 18(3) :161 – 172, 2004.

[175] V. Pareto. Cours d'économie politique. *F. Rouge. Lausanne*, 1 et 2, 1896.

[176] Marcello Pelillo, Fabio Abbattista, and Angelo Maffione. An evolutionary approach to training relaxation labeling processes. *Pattern Recognition Letters*, 16(10) :1069 – 1078, 1995.

[177] D. Peng, R. Jones. An approach based on biological algorithm for three-dimensional shape optimisation with fracture strength constrains. *Comput. Methods Appl. Mech. Engrg.*, xxx :000–000, 2008.

[178] M.-C. Portmann, A. Vignier, D. Dardilhac, and D. Dezalay. Branch and bound crossed with {GA} to solve hybrid flowshops. *European Journal of Operational Research*, 107(2) :389 – 400, 1998.

[179] M.J.D. Powell. Direct search algorithms for optimization calculations. *Acta Numerica*, 7 :287–336, 1998.

[180] M.J.D. Powell. On trust region methods for unconstrained minimization without derivatives. *Mathematical Programming*, 97 :605–623, 2003.

[181] W. Prager, J.E. Taylor. Problems of optimal structural design. *Journal of Applied Mechanics*, 35 :102–106, 1968.

[182] Ph. Preux and E.-G. Talbi. Towards hybrid evolutionary algorithms. *International Transactions in Operational Research*, 6(6) :557 – 570, 1999.

[183] C.J. Price, I.D. Coope, and D. Byatt. A convergent variant of the nelder mead algorithm. *Journal of Optimization Theory and Applications*, 113 :5–19, 2002.

[184] C.J. Price, I.D. Coope. Frames and grids in unconstrained and linearly constrained optimization : A nonsmooth approach. *SIAM Journal on Optimization*, 14 :415–438, 2003.

[185] A. Prékopa. Stochastic programming. *Kluwer Academic Publishers, Dordrecht*, 1995.

[186] S. Qu, K. Zhang, and F. Wang. A global optimization using linear relaxation for generalized geometric programming. *European Journal of Operational Research*, 190 :345–356, 2008.

[187] S. S. Rao. Engineering opimization, theory and practice. *Wiley*, 1996.

[188] H. Ratschek, J. Rokne. New computer methods for global optimization. *Ellis Horwood Limited Market Cross House, Cooper Street, Chichester, West Sussex, PO19 1EB, England*, 1988.

[189] M. Redhe and L. Nilsson. Optimization of the new saab 9-3 exposed to impact load using a space mapping technique. *Struct Multidisc Optim*, 27 :411–420, 2004.

[190] M.J. Rijckaert and X.M. Martens. Comparison of generalized geometric programming algorithms. *Journal of Optimization Theory and Application*, 26 :205–241, 1978.

[191] B. Ritzei, al. Using genetic algorithms to solve a multiple objective groundwater pollution containment problem. *Water Resources Research*, 30 :1589–1603, 1994.

[192] G.I.N. Rozvany, B.L. Karihaloo. Structural optimization. *Kluwer Academic Publi shers, London*, 1988.

[193] T. P. Runarsson and X. Yao. Stochastic ranking for constrained evolutionary optimization. *IEEE Transactions on Evolutionary Computation*, 4 :284–294, 2000.

[194] S. K. Sachdeva, P. B. Nair, and A. J. Keane. On using deterministic fea software to solve problems in stochastic structural mechanics. *Computers and Structures*, 85 :277–290, 2007.

[195] M. Save, W. Prager. Structural optimization : v.1. optimality criteria, v.2. mathematical programming. *Plenum Press, New York*, 1985.

[196] D. Schaffer. Multiple objective optimisation with vector evaluated genetic algorithm. *In genetic Algorithm and their Applications : Proceedings of the First International Conference on Genetic Algonthm*, pages 93–100, 1985.

[197] K. Schittkowski. A fortran subroutine solving constrained nonlinear programming problems. *Annals of Operation Research*, pages 485–500, 1985.

[198] K. Schittkowski. Numerical data fitting in dynamical systems - a practical introduction with applications and software. *Kluwer Academic Publishers, Dordrecht, Boston, London*, 2002.

[199] L.A. Schmit. Structural design by systematic synthesis. *Proceedings of the 2^{nd} ASCE Conference on Electronic Computation, New York*, pages 105–132, 1960.

[200] L.A. Schmit, B. Farshi. Some approximation concepts for structural synthesis. *AIAA Journal*, 12 :692–699, 1974.

[201] L.A. Schmit, H. Miura. Approximation concepts for efficient structural synthesis. *NASA Contractor Report, NASA-CR 2552*, 1976.

[202] M. Schoenauer, S. Xanthakis. Constrained ga optimization. *Proceedings of the fifth International Confrrence on Genetic Algorithms*, 5 :573–580, 1993.

[203] D. F. Shanno. Conditioning of quasi-newton methods for function minimization. *Mathematics of Computation*, 24 :647–656, 1970.

[204] P. S. Shelokar, P. Siarry, V.K. Jayaraman, and B.D. Kulkarni. Particle swarm and ant colony algorithms hybridized for improved continuous optimization. *Applied Mathematics and Computation*, 188 :129–142, 2007.

[205] P. Shen and K. Zhang. Global optimization of signomial geometric programming using linear relaxation. *Applied Mathematics and Computation*, 150 :99–114, 2004.

[206] B. W. Silverman. Density estimation for statistics and data analysis. 1986.

[207] H. Smaoui, C. Fleury, and L.A. Schmit. Advances in dual algorithms and convex approximation methods. *Proceedings of AIAA/ASME/ASCE 29th Structures, Structural Dynamics and Material Conference*, pages 1339–1347, 1988.

[208] F. O. Sonmez. Shape optimization of 2d structures using simulated annealing. *Comput. Methods Appl. Mech. Engrg.*, 196 :3279–3299, 2007.

[209] J. C. Spall. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE transactions on automatic control*, 37 :332–341, 1992.

[210] J. C. Spall. An overview of the simultaneous perturbation method for efficient optimization. *Johns Hopkins apl technical digest*, 19, 1998.

[211] N. Srivinas and K. Deb. Multiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary Computation*, 2 :221–248, 1994.

[212] W. Stadler and J. Duer. Multicriteria optimization in engineering : A tutorial and survy. *In Structural optimization : Status and future. American institute of Aeronautics and Astronautics*, pages 209–249, 1992.

[213] Thomas Stutzle and Holger H. Hoos. Max-min ant system. *Future Generation Computer Systems*, 16(8) :889 – 914, 2000.

[214] K. Svanberg. The method of moving asymptotes-a new method for structural optimization. *international journal for numerical methods in engineering*, 24 :359–373, 1987.

[215] Austrian electronic card contract awarded to german manufacturer. *Card Technology Today*, 16(5) :1 –, 2004.

[216] E.-G. Talbi, S. Cahon, and N. Melab. Designing cellular networks using a parallel hybrid metaheuristic on the computational grid. *Computer Communications*, 30(4) :698 – 713, 2007.

[217] E.-G. Talbi, Z. Hafidi, D. Kebbal, and J.-M. Geib. A fault-tolerant parallel heuristic for assignment problems. *Future Generation Computer Systems*, 14(56) :425 – 438, 1998. <ce :title>Bio-inspired solutions to parallel processing problems</ce :title>.

[218] E.-G. Talbi, O. Roux, C. Fonlupt, and D. Robillard. Parallel ant colonies for the quadratic assignment problem. *Future Generation Computer Systems*, 17(4) :441 – 449, 2001.

[219] E.D. Talbi, L. Geneste, B. Grabot, R. Prévitali, and Pascal Hostachy. Application of optimization techniques to parameter set-up in scheduling. *Computers in Industry*, 55(2) :105 – 124, 2004.

[220] El-Ghazali Talbi and Clarisse Dhaenens. Cooperative combinatorial optimization. *European Journal of Operational Research*, 199(3) :619, 2009.

[221] El-Ghazali Talbi and Geir Hasle. Special issue of the journal of parallel and distributed computing : Metaheuristics on {GPU}. *Journal of Parallel and Distributed Computing*, 71(4) :621 – 622, 2011.

[222] El-Ghazali Talbi and Geir Hasle. Metaheuristics on {GPUs}. *Journal of Parallel and Distributed Computing*, 73(1) :1 – 3, 2013.

[223] M.L. Talbi and A. Charef. {PVC} discrimination using the {QRS} power spectrum and self-organizing maps. *Computer Methods and Programs in Biomedicine*, 94(3) :223 – 231, 2009.

[224] Sarosh N Talukdar. Collaboration rules for autonomous software agents. *Decision Support Systems*, 24(3-4) :269 – 278, 1999.

[225] A.-A. Tantar, N. Melab, E.-G. Talbi, B. Parent, and D. Horvath. A parallel hybrid genetic algorithm for protein structure prediction on the computational grid. *Future Generation Computer Systems*, 23(3) :398 – 409, 2007.

[226] V. T'kindt, J.-C. Billaut, J.-L. Bouquard, C. LentÃ©, P. Martineau, E. NÃ©ron, C. Proust, and C. Tacquard. The e-ocea project : towards an internet decision system for scheduling problems. *Decision Support Systems*, 40(2) :329 – 337, 2005.

[227] M. D. Toksari. Ant colony optimization for finding the global minimum. *Applied Mathematics and Computation*, 176 :308–316, 2006.

[228] V. Torczon. On the convergence of pattern search algorithms. *SIAM Journal on Optimization*, 7 :1–25, 1997.

[229] E. Triki, Y. Collette, and P. Siarry. A theoretical study on the behavior of simulated annealing leading to a new cooling schedule. *European Journal of Operational Research*, 166 :77–92, 2005.

[230] I. G. Tsoulos. Modifications of real code genetic algorithm for global optimization. *Applied Mathematics and Computation*, 203 :598–607, 2008.

[231] M. Valenzuela and E. Uresti-Charre. A nongenerational genetic algorithm for multiobjective optimization. *Proceedings of the Seventh International Conference on Genetic Algorithms*, pages 658–665, 1997.

[232] D. A. Van Veidhuizen. Multiobjective evolutionary algorithms : Classification, analyses and new innovation. *Air Force Institute of Technologv. United States*, 22 :425–460, 1999.

[233] C. Wang, Y. Yang, and J. Li. A new filled function method for unconstrained global optimization. *Journal of Computational and Applied Mathematics*, 225 :68–79, 2009.

[234] L. Wang, K. Chen, and Y.S. Ong. Advances in natural computation. *Part III, Springer Science & Business Publisher, Changsha, China*, 2005.

[235] Y. J. Wang, J. S. Zhang. An efficient algorithm for large scale global optimization of continuous functions. *Journal of Computational and Applied Mathematics*, 206 :1015–1026, 2007.

[236] L. Wei, M. Zhao. A niche hybrid genetic algorithm for global optimization of continuous multimodal functions. *Applied Mathematics and Computation*, 160 :649–661, 2005.

[237] D. Fi. Wolpert, W. G. Macready. No free lunch theoremns for optimnization. *IEEE Transactions on Evolutionary Computation*, 1 :67–82, 1997.

[238] David L Woodruff. Proposals for chunking and tabu search. *European Journal of Operational Research*, 106(2-3) :585 – 598, 1998.

[239] Z. Xinchao. A perturbed particle swarm algorithm for numerical optimization. *Applied Soft Computing*, 10 :119–124, 2010.

[240] Y. W. Yang, J. F. Xu, and C. K. Soh. An evolutionary programming algorithm for continuous global optimization. *European Journal of Operational Research*, 168 :354–369, 2006.

[241] Q. Yuan, Z. He, and H. Leng. A hybrid genetic algorithm for a class of global optimization problems with box constraints. *Applied Mathematics and Computation*, 197 :924–929, 2008.

[242] C. Zhang, J. Ning, S. Lu, D. Ouyang, and T. Ding. A novel hybrid differential evolution and particle swarm optimization algorithm for unconstrained optimization. *Operations Research Letters*, 37 :117–122, 2009.

[243] E. Zitzler, M. Laumanns, and L. Thiele. Spea2 : Improving the strength pareto evolutionary algorithm. *Swiss Federal Institute of Technology, Lausanne, Switzerlnd, Tech. Rep. TIK-Rep*, 103, 2001.

[244] E. Zitzler, L. Thiele. An evolutionaryalgorithrn for multiobjective optirnization : The strength pareto approach. *TIK Report n˚43*, 1998.

[245] E. Zitzler, L. Thiele. Multiobjective evolutionary algorithms : a comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computatio*, pages 257–271, 1999.