Perfectionnement d'un algorithme adaptatif d'optimisation par essaim particulaire : application en génie médical et en électronique

Yann Cooren

THÈSE DE DOCTORAT
DE L'UNIVERSITÉ PARIS 12 VAL DE MARNE

par

Yann COOREN

pour obtenir le grade de

DOCTEUR EN SCIENCES

Spécialité : **SCIENCES DE L'INGÉNIEUR**

Option : Optimisation

Équipe d'accueil : Laboratoire Images, Signaux et Systèmes Intelligents (LiSSi, E.A. 3956)
École Doctorale : École Doctorale Sciences et Ingénierie : Matériaux - Modélisation - Environnement (SIMME)

Sujet de la thèse:

Perfectionnement d'un algorithme adaptatif d'Optimisation par Essaim Particulaire. Applications en génie médical et en électronique

---

soutenue le 27 novembre 2008 devant la commission d'examen composée de :

M. Michel GOURGAND	Professeur des Universités	Université de Clermont-Ferrand	**Rapporteur**
M. Laurent KRÄHENBÜHL	Directeur de recherche CNRS	Laboratoire Ampère, Lyon	**Rapporteur**
M. Laurent GENESTE	Professeur des Universités	ENIT, Tarbes	**Examinateur**
M. Ioan Cristian TRELEA	Maître de Conférences HDR	INA-Grignon	**Examinateur**
M. Maurice CLERC	Ingénieur consultant	Annecy	**Examinateur**
M. Patrick SIARRY	Professeur des Universités	Université de Paris 12	**Directeur de thèse**

Résumé

Les métaheuristiques sont une famille d'algorithmes stochastiques destinés à résoudre des problèmes d'"optimisation difficile". Utilisées dans de nombreux domaines, ces méthodes présentent l'avantage d'être généralement efficaces, sans pour autant que l'utilisateur ait à modifier la structure de base de l'algorithme qu'il utilise. Parmi celles-ci, l'Optimisation par Essaim Particulaire (OEP) est une nouvelle classe d'algorithmes proposée pour résoudre les problèmes à variables continues. Les algorithmes d'OEP s'inspirent du comportement social des animaux évoluant en essaim, tels que les oiseaux migrateurs ou les poissons. Les "particules" d'un même essaim communiquent de manière directe entre elles tout au long de la recherche pour construire une solution au problème posé, en s'appuyant sur leur expérience collective.

Reconnues depuis de nombreuses années pour leur efficacité, les métaheuristiques présentent des défauts qui rebutent encore certains utilisateurs. Le réglage des paramètres des algorithmes est un de ceux-ci. Il est important, pour chaque problème posé, de trouver le jeu de paramètres qui conduise à des performances optimales de l'algorithme. Cependant, cette tâche est fastidieuse et coûteuse en temps, surtout pour les utilisateurs novices. Pour s'affranchir de ce type de réglage, des recherches ont été menées pour proposer des algorithmes dits "adaptatifs". Avec ces algorithmes, les valeurs des paramètres ne sont plus figées, mais sont modifiées, en fonction des résultats collectés durant le processus de recherche. Dans cette optique-là, Maurice Clerc a proposé TRIBES, qui est un algorithme d'OEP mono-objectif sans aucun paramètre de contrôle. Cet algorithme fonctionne comme une "boîte noire", pour laquelle l'utilisateur n'a qu'à définir le problème à traiter et le critère d'arrêt de l'algorithme.

Nous proposons dans cette thèse une étude comportementale de TRIBES, qui permet d'en dégager les principales qualités et les principaux défauts. Afin de corriger certains de ces défauts, deux modules ont été ajoutés à TRIBES. Une phase d'initialisation régulière est insérée, afin d'assurer, dès le départ de l'algorithme, une bonne couverture de l'espace de recherche par les particules. Une nouvelle stratégie de déplacement, basée sur une hybridation avec un algorithme à estimation de distribution, est aussi définie, afin de maintenir la diversité au sein de l'essaim, tout au long du traitement. Le besoin croissant de méthodes de résolution de problèmes multiobjectifs a conduit les concepteurs à adapter leurs méthodes pour résoudre ce type de problème. La complexité de cette opération provient du fait que les objectifs à optimiser sont souvent contradictoires. Nous avons élaboré une version multiobjectif de TRIBES, dénommée MO-TRIBES. Nos algorithmes ont été enfin appliqués à la résolution de problèmes de seuillage d'images médicales et au problème de dimensionnement de composants de circuits analogiques.

Mots-clefs : Optimisation, optimisation continue, optimisation multiobjectif, métaheuristiques, optimisation par essaim particulaire, algorithme adaptatif, algorithme à estimation de distribution, segmentation d'image, seuillage.

Abstract

Metaheuristics are a new family of stochastic algorithms which aim at solving "difficult" optimization problems. Used to solve various applicative problems, these methods have the advantage to be generally efficient on a large amount of problems. Among the metaheuristics, Particle Swarm Optimization (PSO) is a new class of algorithms proposed to solve continuous optimization problems. PSO algorithms are inspired from the social behavior of animals living in swarm, such as bird flocks or fish schools. The "particles" of the swarm use a direct way of communication in order to build a solution to the considered problem, based on their collective experience.

Known for their efficiency, metaheuristics show the drawback of comprising too many parameters to be tuned. Such a drawback may rebuff some users. Indeed, according to the values given to the parameters of the algorithm, its performance fluctuates. So, it is important, for each problem, to find the parameter set which gives the best performance of the algorithm. However, such a problem is complex and time consuming, especially for novice users. To avoid the user to tune the parameters, numerous researches have been done to propose "adaptive" algorithms. For such algorithms, the values of the parameters are changed according to the results previously found during the optimization process. TRIBES is an adaptive mono-objective parameter-free PSO algorithm, which was proposed by Maurice Clerc. TRIBES acts as a "black box", for which the user has only the problem and the stopping criterion to define. The first objective of this PhD is to make a global study of the behavior of TRIBES under several conditions, in order to determine the strengths and drawbacks of this adaptive algorithm. In order to improve TRIBES, two new strategies are added. First, a regular initialization process is defined in order to insure an exploration as wide as possible of the search space, since the beginning of the optimization process. A new strategy of displacement, based on an hybridation with an estimation of distribution algorithm, is also introduced to maintain the diversity in the swarm all along the process. The increasing need for multiobjective methods leads the researchers to adapt their methods to the multiobjective case. The difficulty of such an operation is that, in most cases, the objectives are conflicting. We designed MO-TRIBES, which is a multiobjective version of TRIBES. Finally, our algorithms are applied to thresholding segmentation of medical images and to the design of electronic components.

Keywords : Optimization, continuous optimization, multiobjective optimization, metaheuristics, particle swarm optimization, adaptive algorithm, estimation of distribution algorithm, image segmentation, thresholding.

Table des matières

Introduction générale

Cette thèse part du constat simple que les problèmes d'optimisation occupent une place de plus en plus importante dans les processus décisionnels. Dans de multiples domaines, le besoin toujours croissant d'efficacité et de rentabilité a conduit les décisionnaires à ne plus se fier seulement à l'expérience des experts, mais aussi aux résultats numériques obtenus par la résolution de problèmes d'optimisation. Une telle émulation a donné naissance à de nouvelles méthodes de résolution de problèmes d"'optimisation difficile".

L'étude des phénomènes réels est une source d'inspiration en ingénierie informatique, où l'étude et la modélisation des systèmes complexes sont très présentes. En recherche opérationnelle, et plus précisément dans le domaine de l'optimisation, de nombreuses méthodes sont inspirées par de telles études. Parmi ces domaines d'inspiration, la biologie est particulièrement prisée par les chercheurs en optimisation. En effet, la modélisation des systèmes "intelligents" rencontrés dans la nature peut servir à créer une "intelligence artificielle", à même de résoudre des problèmes d'optimisation.

Parmi toutes les branches de la biologie, l'éthologie, étude du comportement des animaux, a récemment donné lieu à plusieurs avancées significatives dans le domaine de la recherche opérationnelle. La modélisation des comportements animaliers, principalement ceux des animaux évoluant en essaim, est exploitée majoritairement en robotique, en classification ou encore en optimisation. Un système en essaim est un système qui doit son intelligence à la coopération d'éléments qui, individuellement, sont de faible intelligence. La performance du système entier est supérieure à la somme des performances de ses parties. Dans le cadre de l'optimisation, cette approche a donné lieu à la naissance de nouvelles "métaheuristiques". Les métaheuristiques forment une famille d'algorithmes stochastiques destinée à la résolution de problèmes d"'optimisation difficile". Ces méthodes ont été conçues afin de résoudre un panel très large de problèmes, sans pour autant nécessiter de changements profonds dans les structures des algorithmes. Ces méthodes sont inspirées par des analogies avec la physique (recuit simulé), la génétique (algorithmes évolutionnaires) et l'éthologie (colonies de fourmis). Parmi les méthodes inspirées de l'éthologie, la méthode d'Optimisation par Essaim Particulaire (OEP) a été proposée en 1995 par Kennedy et Eberhart pour la résolution de problèmes d'optimisation difficile continus. Bénéficier d'un algorithme d'optimisation difficile à variables continues est un avantage étant donné que la plupart des méthodes existantes ont été conçues pour traiter les problèmes discrets, et que les problèmes à variables continues sont très courants en ingénierie.

L'OEP présente, comme la majorité des métaheuristiques, le défaut de nécessiter le réglage de plusieurs paramètres de contrôle. Les paramètres de contrôle permettent de "régler" le fonctionnement de l'algorithme en fonction du problème proposé. Les performances de l'OEP sont fortement corrélées au réglage de ses paramètres. Il est donc indispensable pour un concepteur d'étudier l'influence de chaque paramètre sur le comportement de l'algorithme. De cette étude découlera le choix du jeu de paramètres optimal. Idéalement, il serait nécessaire de mener cette étude pour chaque nouveau problème, afin d'adapter au mieux le comportement de la méthode au problème posé.

Cette stratégie apparaît compliquée à mettre en place dans le cadre de l'industrie. En effet, un ingénieur qui doit résoudre un problème d'optimisation n'a pas le temps de tester de nombreux

jeux de paramètres avant de décider pour lequel opter. Le besoin de résultats étant immédiat, il est important pour l'ingénieur de disposer de méthodes qui donnent des résultats satisfaisants pour une large gamme de problèmes, sans pour autant qu'il soit nécessaire de devoir modifier le jeu de paramètres. Ce constat a conduit au développement de méthodes dites "adaptatives". Ces méthodes proposent de réduire le nombre de paramètres d'un algorithme, en définissant des règles d'adaptation, qui permettent de modifier la valeur des paramètres, en fonction des résultats trouvés au cours du déroulement de l'algorithme. L'idéal serait de concevoir des algorithmes qui ne nécessitent aucun paramètre de contrôle. Dans ce cas, l'utilisateur n'aurait qu'à définir le problème posé et le critère d'arrêt de l'algorithme. Cependant, le gain de temps lié à l'automatisation de la définition des paramètres ne doit pas être la cause d'une perte d'efficacité de l'algorithme. La réduction du nombre de paramètres des méthodes d'optimisation est aujourd'hui un enjeu crucial. En effet, les applications sont nombreuses et il existe aujourd'hui beaucoup de méthodes qui se sont révélées efficaces. Cependant, ces méthodes restent toujours difficiles à régler de manière "optimale", pour un utilisateur non spécialiste.

Cette thèse a été préparée au sein du *Laboratoire Images, Signaux et Systèmes Intelligents* (LiSSi, E.A. 3956) de l'Université de Paris 12 Val de Marne. Ce laboratoire est en partie spécialisé dans l'imagerie et le traitement du signal pour le génie biomédical. Cette thèse a été financée par une allocation de recherche du Ministère de la Recherche ainsi que par un poste de moniteur à l'Université de Paris12 Val de Marne. Ce travail a été encadré par le Professeur P. Siarry, directeur du laboratoire et responsable du groupe "optimisation", et co-encadré par M. Clerc, spécialiste de la méthode d'OEP et consultant pour le projet XPS (*eXtended Particle Swarm*). Le groupe "optimisation" est une équipe transversale au LiSSi spécialisée dans l'adaptation des métaheuristiques aux problèmes à variables continues, mais aussi dans l'optimisation multiobjectif ou dynamique. Ont participé à mon travail de doctorat Amir Nakib, docteur de l'Université de Paris12 Val de Marne, et Mourad Fakhfakh, Maître de Conférences à l'École Nationale d'Ingénieurs de Sfax, en Tunisie.

Le but de cette thèse est l'étude de TRIBES, un algorithme d'Optimisation par Essaim Particulaire sans paramètres proposé par M. Clerc. L'objectif premier est d'étudier le comportement de TRIBES sur un large panel de fonctions de test. L'étude des résultats numériques et du comportement des particules, tout au long du traitement, permet de mettre en évidence les avantages et les défauts de TRIBES. Le deuxième objectif de cette thèse est de proposer des méthodes alternatives, incorporables à TRIBES, qui permettent d'en améliorer l'efficacité. Enfin, devant la nécessité croissante de disposer de méthodes multiobjectifs, la dernière contribution de cette thèse est de proposer une version multiobjectif de TRIBES, dénommée MO-TRIBES. Comme sa version mono-objectif, MO-TRIBES possède la particularité de ne posséder aucun paramètre de contrôle à régler.

Les applications étudiées ont porté sur l'utilisation de TRIBES pour résoudre des problèmes de traitement d'images et de dimensionnement de circuits électroniques. Dans le cas de la segmentation d'images médicales, TRIBES est utilisé comme méthode d'optimisation pour maximiser un critère qui permet de déterminer les différents seuils d'une image multi-classe. Dans le cas de l'application en électronique, TRIBES est utilisé dans sa version multiobjectif, pour dimensionner les transistors d'un amplificateur faible bruit et d'un convoyeur de courant de seconde génération, afin que ceux-ci présentent des performances optimales.

Ce manuscrit se divise en trois parties. La première partie est consacrée à l'étude bibliographique des méthodes d'optimisation. Elle fait l'objet du premier chapitre. La deuxième partie est consacrée à TRIBES, dans ses versions mono-objectif et multiobjectif. Elle fait l'objet des chapitres 2 et 3. Enfin, la dernière partie correspond aux applications de TRIBES à des problèmes "réels". Les applications concernent les chapitres 4 et 5.

Dans le chapitre 1, nous présentons un état de l'art des métaheuristiques d'optimisation mono-objectif ou multiobjectif. Une attention toute particulière est accordée à la méthode d'Optimisation par Essaim Particulaire, principale méthode utilisée au cours de cette thèse. Nous abordons aussi l'intérêt des méthodes adaptatives. Un état de l'art des algorithmes d'Optimisation par Essaim Particulaire adaptatifs est proposé.

Le chapitre 2 est consacré à l'étude de TRIBES dans sa version mono-objectif. Une étude détaillée de TRIBES est réalisée sur un large éventail de problèmes, afin de dégager les avantages et les inconvénients de cet algorithme. Cette étude est basée sur les résultats numériques de TRIBES, mais aussi sur l'analyse du comportement des particules au cours de l'exécution de l'algorithme. Une comparaison de TRIBES avec d'autres algorithmes est présentée, afin de montrer en quoi TRIBES apparaît comme un algorithme prometteur. La comparaison est faite en utilisant la procédure de test définie lors de la conférence *2005 International Conference on Evolutionary Computation*. Enfin, deux améliorations sont proposées pour améliorer le comportement de TRIBES. La première amélioration est une nouvelle méthode d'initialisation de TRIBES, qui permet de préserver la diversité de l'essaim, en essayant de couvrir au maximum l'espace de recherche, dès l'initialisation. La seconde amélioration est la définition d'une nouvelle stratégie pour les particules, qui utilise une hybridation avec un Algorithme à Estimation de Distribution.

Le chapitre 3 est consacré à la présentation de MO-TRIBES, qui est la version multiobjectif de TRIBES. Cet algorithme, développé au cours de cette thèse, est un algorithme d'optimisation multiobjectif à variables continues sans paramètres. Le but est d'adapter les règles de TRIBES, afin de pouvoir traiter des problèmes multiobjectifs. Cet algorithme utilise une approche Pareto en incorporant à TRIBES la gestion d'une archive externe. Cette archive permet de stocker toutes les solutions "non-dominées" et, ainsi, de fournir, en fin de traitement, une approximation du front de Pareto du problème traité. MO-TRIBES a été créé en respectant la contrainte qui imposait de ne devoir introduire aucun paramètre de contrôle. Une comparaison de MO-TRIBES avec les algorithmes les plus connus sur des problèmes classiques est présentée en fin de chapitre.

Dans le chapitre 4, nous présentons une première application de TRIBES dans le cas mono-objectif. Le but est de segmenter des images multi-classe dont le nombre de classes est connu a priori. A cet effet, on cherche à déterminer automatiquement les seuils des différentes classes. TRIBES est utilisé pour optimiser deux types de critères qui permettent de déterminer ces seuils. Le premier critère utilisé est l'entropie exponentielle à deux dimensions, qui permet de déterminer directement les seuils de différentes classes, l'entropie étant maximale pour les valeurs optimales des seuils. La deuxième approche est l'approximation de l'histogramme de l'image par une somme de gaussiennes. Approcher l'histogramme de l'image par une somme de gaussiennes permet de connaître l'expression analytique approchée de l'histogramme, donc de calculer beaucoup plus facilement les seuils des différentes classes.

Le chapitre 5 présente des applications de TRIBES et MO-TRIBES au dimensionnement de circuits électroniques. Le but est de trouver les dimensions des transistors qui permettent de faire fonctionner de manière optimale les circuits étudiés. En d'autres termes, notre objectif est de déterminer la largeur et la longueur des transistors qui permettent au circuit d'avoir des performances optimales, suivant un critère donné. TRIBES est utilisé pour maximiser le gain d'un amplificateur faible bruit. MO-TRIBES est utilisé pour minimiser la résistance d'entrée et maximiser la fréquence de transition d'un convoyeur de courant de seconde génération.

Enfin, la conclusion récapitule les contributions de cette thèse et propose des perspectives aux travaux effectués.

Chapitre 1

Métaheuristiques d'optimisation : état 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. Les problèmes d'apprentissage de réseaux de neurones, de planification des tâches ou encore d'identification sont, par exemple, des problèmes communs d'optimisation.

Ce chapitre présente un état de l'art des métaheuristiques d'optimisation aussi bien pour les problèmes mono-objectif que pour les problèmes multiobjectifs. Un intérêt particulier est apporté à la méthode d'Optimisation par Essaim Particulaire (OEP), qui constitue le sujet principal de ce travail de thèse. De plus, nous abordons, dans le cas mono-objectif, la question de l'intérêt grandissant des méthodes adaptatives. Un bref aperçu des OEPs adaptatives est présenté dans ce chapitre.

L'Optimisation par Essaim Particulaire est une nouvelle classe des métaheuristiques proposée en 1995 par Kennedy et Eberhart [Kennedy et al., 1995]. Cet algorithme s'inspire du comportement social des animaux évoluant en essaim. Les différentes particules d'un essaim communiquent directement avec leurs voisins et construisent ainsi une solution à un problème, en s'appuyant sur leur expérience collective.

1.2 Optimisation mono-objectif

1.2.1 Optimisation mono-objectif difficile

Un problème d'optimisation en général est défini par un espace de recherche S et une fonction objectif f. Le but est de trouver la solution $s^* \in S$ de meilleure qualité $f(s^*)$. Suivant le problème posé, on cherche soit le minimum soit le maximum de la fonction f. Dans la suite de ce document, nous aborderons les problèmes d'optimisation essentiellement sous l'aspect minimisation, maximiser une fonction f étant équivalent à minimiser $-f$. L'équation (1.1) résume la définition précédente.

$$s^* = \min\left(f(s) \setminus s \in S\right) \tag{1.1}$$

De plus, un problème d'optimisation peut présenter des contraintes d'égalité et/ou d'inégalité sur les solutions candidates $s \in S$, être multiobjectif si plusieurs fonctions objectifs doivent être

optimisées ou encore dynamique, si la "topologie" de f change au cours du temps. Dans cette section, nous n'étudierons que les problèmes mono-objectif, le cas multiobjectif étant étudié section 1.3.

Il existe de nombreuses méthodes déterministes (ou "exactes") qui permettent de résoudre certains types de problèmes d'optimisation en un temps fini. Cependant, ces méthodes nécessitent que la fonction objectif présente un certain nombre de caractéristiques, telles que la convexité, la continuité ou encore la dérivabilité. Parmi les méthodes les plus connues, on peut citer les méthodes de programmation linéaire [Schrijver, 1998], quadratique [Nocedal et al., 1999] ou dynamique [Bertsekas, 2000], la méthode de Newton [Nocedal et al., 1999], la méthode du "simplex" [Nelder et al., 1965] ou encore la méthode du gradient [Avriel, 2003].

Certains problèmes restent cependant trop complexes à résoudre pour les méthodes déterministes. Certaines caractéristiques peuvent être problématiques pour ces méthodes. Parmi celles-ci, on peut citer la discontinuité, la non-dérivabilité ou encore la présence de bruit. Dans ce cas, le problème d'optimisation est dit *difficile,* car aucune méthode déterministe ne peut résoudre ce problème en un temps "raisonnable".

Les problèmes d'optimisation difficile se divisent en deux catégories : les problèmes à variables discrètes et les problèmes à variables continues. De façon générale, un problème d'optimisation à variables discrètes, ou combinatoire, consiste à trouver, dans un ensemble discret, la meilleure solution réalisable, au sens du problème défini par (1.1). Le problème majeur réside ici dans le fait que le nombre de solutions réalisables est généralement très élevé, donc il est très difficile de trouver la meilleure solution dans un temps "raisonnable". Le problème du voyageur de commerce [Biggs et al., 1976] est un exemple classique de problème d'optimisation à variables discrètes. L'utilisation d'algorithmes d'optimisation stochastiques, tels que les métaheuristiques, permet de trouver une solution approchée, en un temps "raisonnable".

Les problèmes à variables continues sont, eux, moins formalisés que les précédents. En effet, la grande majorité des métaheuristiques existantes ont été créées pour résoudre des problèmes à variables discrètes [Siarry et al., 2006]. Cependant, la nécessité croissante de méthodes pouvant résoudre ce type de problèmes a poussé les chercheurs à adapter leurs méthodes au cas continu. Les difficultés les plus courantes de ce type de problèmes sont les corrélations non identifiées entre les variables, le caractère bruité des fonctions ou encore des fonctions objectifs qui ne sont accessibles que par dispositif expérimental. Il est à noter qu'il existe des problèmes à variables mixtes, c'est-à-dire que le problème présente à la fois des variables discrètes et continues.

1.2.2 Métaheuristiques pour l'optimisation mono-objectif difficile

Les métaheuristiques sont une famille d'algorithmes stochastiques destinés à la résolution des problèmes d'optimisation. Leur particularité réside dans le fait que celles-ci sont adaptables à un grand nombre de problèmes sans changements majeurs dans leurs algorithmes, d'où le qualificatif *méta*. Leur capacité à optimiser un problème à partir d'un nombre minimal d'informations est contrebalancée par le fait qu'elles n'offrent aucune garantie quant à l'optimalité de la meilleure solution trouvée. Seule une approximation de l'optimum global est donnée. Cependant, du point de vue de la recherche opérationnelle, ce constat n'est pas forcément un désavantage, étant donné que l'on préférera toujours une approximation de l'optimum global trouvée rapidement qu'une valeur exacte trouvée dans un temps rédhibitoire.

Les métaheuristiques sont des méthodes qui ont, en général, un comportement itératif, c'est-à-dire que le même schéma est reproduit un certain nombre de fois au cours de l'optimisation, et qui sont "directes", dans le sens où elles ne font pas appel au calcul du gradient de la fonction. Ces méthodes tirent leur efficacité du fait qu'elles sont moins facilement piégeables dans des optima locaux, car elles acceptent, au cours du traitement, des dégradations de la fonction objectif et la

recherche est souvent menée par une population de points et non un point unique.

Comme il a été dit section 1.1, les métaheuristiques sont majoritairement conçues pour résoudre des problèmes à variables discrètes, mais font de plus en plus l'objet d'adaptations aux problèmes à variables continues. De plus, de par leur variété et leur capacité à résoudre des problèmes très divers, les métaheuristiques sont assez facilement sujettes à extensions. Parmi celles-ci, on peut citer :

- Les métaheuristiques pour l'optimisation multiobjectif. La but est ici non pas de trouver un optimum global mais de trouver un ensemble d'optima qui forment une surface de compromis pour les différents objectifs du problème. Cette partie sera traitée section 1.3 ;
- Les métaheuristiques pour l'optimisation multimodale [Goldberg et al., 1987], où l'on ne cherche plus l'optimum global, mais l'ensemble des meilleurs optima locaux ;
- Les métaheuristiques pour l'optimisation dynamique [Branke, 2001 ; Ourique et al., 2002 ; Di Caro et al., 1998], où il faut approcher l'optimum à chaque pas de temps, car la fonction objectif change de topologie au cours du temps ;
- Les métaheuristiques hybrides [Talbi, 2002], qui combinent différentes métaheuristiques, afin d'en tirer les avantages respectifs ;
- Les métaheuristiques parallèles [Alba, 2005], pour lesquelles on cherche à accélérer le calcul, en distribuant la charge de calcul sur plusieurs calculateurs.

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 leur réglages, afin de les adapter au mieux aux problèmes posés.

Le monde des métaheuristiques est un monde en constante évolution. De nombreuses méthodes sont proposées chaque année pour tenter d'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 évolutionnaires, les algorithmes de colonies et fourmis ou encore l'optimisation par essaim particulaire. La suite de ce chapitre est dédié à la présentation de ces méthodes.

1.2.2.1 Algorithme de recuit simulé

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 élevées, avant de le laisser refroidir lentement. Ce processus est appelé le *recuit*.

Le recuit simulé a été développé simultanément par Kirkpatrick et al. [Kirpatrick et al., 1983] et Cerny [Cerny, 1985]. Le recuit simulé repose sur l'algorithme de Metropolis [Metropolis et al., 1953 ; Hastings, 1970] décrit par l'Algorithme 1. Cette procédure permet de sortir des minima locaux avec une probabilité élevée si la température T est élevée et, quand l'algorithme atteint de très basses températures, de conserver les états les plus probables.

L'algorithme de Metropolis permet d'échantillonner la fonction objectif par le biais d'une distribution de Boltzmann de paramètre T. Le point crucial de l'algorithme est donc la loi de décroissance de la température. De nombreuses lois de décroissance différentes ont été proposés [Triki et al., 2005].

L'algorithme de recuit simulé est résumé par l'Algorithme 2.

L'algorithme de recuit simulé est devenu rapidement populaire, du fait de sa facilité d'adaptation à un grand nombre de problèmes et son efficacité. En revanche, cet algorithme présente l'inconvénient de disposer d'un nombre élevé de paramètres (température initiale, règle de décroissance de la température, durée des paliers de température, etc.) qui rendent les réglages de l'algorithme assez empiriques. Cependant, il existe des études qui s'attachent à régler de manière optimale les para-

Algorithme 1 Algorithme de Metropolis

Initialiser un point de départ x_0 et une température T
Pour $i = 1$ à n **faire**
  Tant que x_i n'est pas accepté **faire**
    Si $f(x_i) \leq f(x_{i-1})$: accepter x_i
    Si $f(x_i) > f(x_{i-1})$: accepter x_i avec la probabilité $e^{-\frac{f(x_i)-f(x_{i-1})}{T}}$
  Fin Tant que
Fin Pour

Algorithme 2 Algorithme de recuit simulé

Déterminer une configuration aléatoire S
Choix des mécanismes de perturbation d'une configuration
Initialiser la température T
Tant que la condition d'arrêt n'est pas atteinte **faire**
  Tant que l'équilibre n'est pas atteint **faire**
    Tirer une nouvelle configuration S'
    Appliquer la règle de Metropolis
    Si $f(S') < f(S)$
      $S_{min} = S'$
      $f_{min} = f(S')$
    Fin Si
  Fin Tant que
  Décroître la température
Fin Tant que

Algorithme 3 Algorithme de Recherche Tabou

Déterminer une configuration aléatoire s
Initialiser une liste tabou vide
Tant que le critère d'arrêt n'est pas atteint **faire**
  Perturbation de s suivant N mouvements non tabous
  Évaluation des N voisins
  Sélection du meilleur voisin t
  Actualisation de la meilleure position connue s^*
  Insertion du mouvement $t \rightarrow s$ dans la liste tabou
  $s = t$
Fin Tant que

mètres de l'algorithme [Siarry, 1994]. L'autre inconvénient majeur de cette méthode est sa lenteur. Pour pallier ce problème, plusieurs méthodes de parallélisation des calculs ont été introduites [Azencott, 1992]. Bien qu'initialement créée pour fonctionner avec des variables discrètes, la méthode du recuit simulé possède des versions continues [Siarry, 1994 ; Courat et al., 94].

1.2.2.2 Recherche Tabou

L'algorithme de Recherche Tabou a été introduit par Glover en 1986 [Glover, 1986]. Le but de cette méthode est d'inculquer aux méthodes de recherche locale un surcroît d'intelligence. L'idée ici est d'ajouter au processus de recherche une mémoire qui permette de mener une recherche plus "intelligente" dans l'espace des solutions. Comme l'algorithme de 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 un 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. L'algorithme modélise ainsi une forme primaire de mémoire à court terme. L'algorithme de recherche tabou peut être résumé par l'Algorithme 3.

Dans sa forme de base, l'algorithme de recherche tabou présente l'avantage de comporter moins de paramètres que l'algorithme de recuit simulé. Cependant, l'algorithme n'étant pas toujours performant, il est souvent approprié de lui ajouter des processus d'intensification et/ou de diversification, qui introduisent de nouveaux paramètres de contrôle [Glover et al., 1997].

1.2.2.3 Algorithmes évolutionnaires

Les algorithmes évolutionnaires, élaborés au cours des années 1950 [Fraser, 1957], forment une famille d'algorithmes de recherche inspirés de l'évolution biologique des espèces. L'idée ici est de s'inspirer de la théorie darwiniste de sélection naturelle pour résoudre des problèmes d'optimisation. Au cours des années 1970, avec l'avènement des calculateurs de forte puissance, de nombreuses tentatives ont été réalisées pour modéliser cette approche. Trois approches se sont détachées :

- Les *stratégies d'évolution* [Rechenberg, 1965 ; Beyer, 2001] qui ont été conçues comme une méthode d'optimisation à variables continues ;
- La *programmation évolutionnaire* [Fogel et al., 1966], qui visait à faire évoluer les structures d'automates à états finis par des successions de croisements et de mutations ;
- Les *algorithmes génétiques* [Holland, 1973 ; Goldberg, 1989 ; Holland, 1992], qui ont été conçus pour résoudre des problèmes d'optimisation à variables discrètes.

Algorithme 4 Algorithme évolutionnaire générique

Initialisation de la population de μ individus
Évaluation des μ individus
Tant que le critère d'arrêt n'est pas atteint **faire**
  Sélection de λ individus en vue de la phase de reproduction
  Croisement des λ individus sélectionnés
  Mutation des λ individus sélectionnés
  Évaluation des λ' enfants obtenus
  Sélection pour le remplacement
Fin Tant que

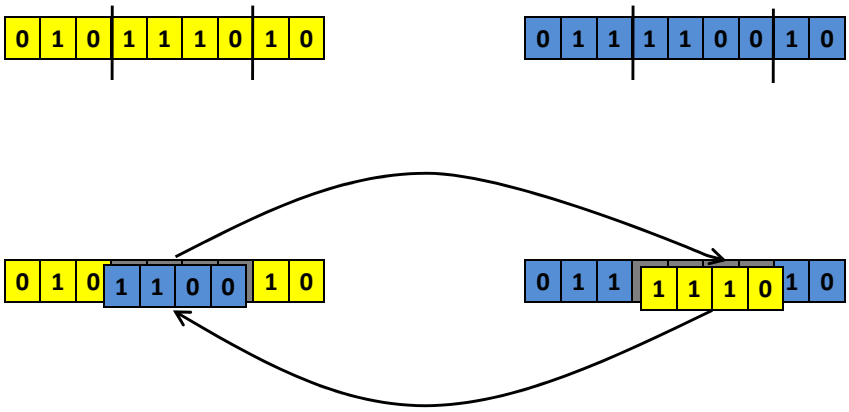


FIGURE 1.1 – Exemple d'opérateur de croisement.

Les approches évolutionnaires s'appuient toutes sur un modèle commun présenté par l'Algorithme 4. Les individus soumis à l'évolution sont des solutions possibles du problème posé. L'ensemble de ces individus constitue une population. Cette population évolue durant une succession d'itérations, appelées générations. Au cours de chaque génération, une succession d'opérateurs est appliquée aux individus de la population pour engendrer une nouvelle population, en vue de la génération suivante. Chaque opérateur utilise un ou plusieurs individus de la population appelé(s) *parent(s)* pour engendrer de nouveaux individus, appelés *enfants*. A la fin de chaque génération, un certain nombre d'individus de la population sont remplacés par des enfants créés durant la génération.

Un algorithme évolutionnaire dispose donc de trois opérateurs principaux : un opérateur de sélection, un opérateur de croisement et un opérateur de mutation. L'opérateur de sélection permet de favoriser la propagation des meilleures solutions dans la population, tout en maintenant la diversité génétique de celle-ci. L'opérateur de croisement est mis en oeuvre lors de la phase de création des enfants. Le but de cet opérateur est d'échanger les gènes des différents parents pour créer les enfants. La Figure 1.1 présente l'exemple d'un croisement simple pour des individus codés en représentation binaire.

L'opérateur de mutation consiste à tirer aléatoirement une composante de l'individu parent et à la remplacer par une valeur aléatoire. L'opérateur de mutation apporte un caractère aléatoire à la création de la descendance, qui permet de maintenir une certaine diversité dans la population. La Figure 1.2 montre un exemple de mutation pour un individu codé en représentation binaire. Une liste

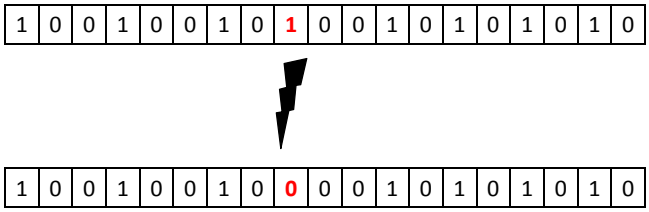


FIGURE 1.2 – Exemple d'opérateur de mutation.

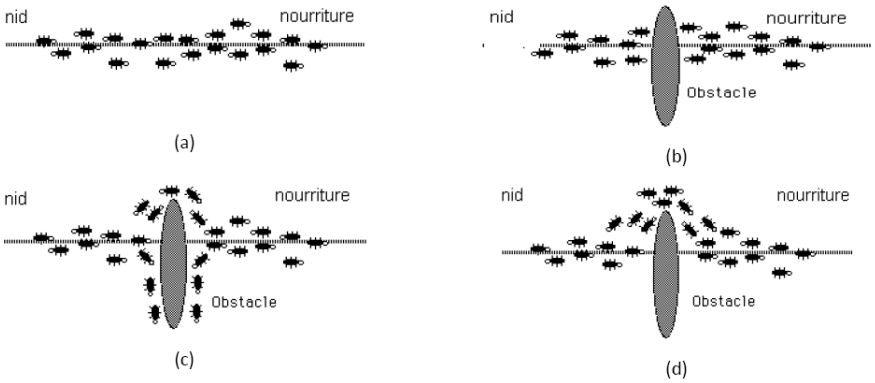


FIGURE 1.3 – Détermination du plus court chemin par une colonie de fourmis. (a) Situation initiale, (b) Introduction d'un obstacle, (c) Recherche du chemin optimal, (d) Prédominance du chemin optimal.

complète de toutes les méthodes existantes pour définir ces opérateurs est disponible dans [Eiben et al., 2003].

1.2.2.4 Algorithmes de colonies de fourmis

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. Ce type de communication indirecte est appelé *stigmergie*.

La principale illustration de ce constat est donnée par la Figure 1.3. On voit sur cette figure que, si un obstacle est introduit sur le chemin des fourmis, les fourmis vont, après une phase de recherche, avoir tendance à toutes emprunter le plus court chemin entre le nid et l'obstacle [Goss et al., 1989]. Plus le taux de phéromone à un endroit donné est important, plus une fourmi va avoir tendance 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 emprunté 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 plus long. De ce fait, le plus court chemin a une probabilité plus grande d'être emprunté par les fourmis que les autres chemins et sera donc, à terme, emprunté par toutes les fourmis.

Le premier algorithme s'inspirant de cette analogie a été proposé par Colorni, Dorigo et Maniezzo

Algorithme 5 Algorithme de colonies de fourmis pour le problème du voyageur de commerce

Tant que le critère d'arrêt n'est pas atteint **faire**
  Pour k=1 à m **faire**
    Choisir une ville au hasard
    Pour chaque ville non visitée i **faire**
      Choisir une ville j, dans la liste J_i^k des villes restantes selon (1.2)
    Fin Pour
    Déposer une piste $\Delta\tau_{ij}^k(t)$ sur le trajet $T^k(t)$ conformément à (1.3)
  Fin Pour
  Évaporer les pistes selon (1.4)
Fin Tant que

[Colorni et al., 1992 ; Dorigo et al., 1996]. Le but initial de la méthode était de résoudre le problème du voyageur de commerce. L'Algorithme 5 présente la méthode proposée par les 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 $\tau_{ij}(t)$, appelée *intensité de la piste*. Cette quantité définit l'attractivité d'une piste et est modifiée après le passage d'une fourmi. C'est la pseudo-mémoire du système.

La règle de déplacement est la suivante [Dorigo et al., 2005] :

$$p_{ij}^k(t) = \begin{cases} \frac{(\tau_{ij}(t))^\alpha (\eta_{ij})^\beta}{\sum_{l \in J_i^k} (\tau_{il}(t))^\alpha (\eta_{il})^\beta} & si\ j \in J_i^k \\ 0 & si\ j \notin J_i^k \end{cases} \tag{1.2}$$

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 est définie par :

$$\Delta\tau_{ij}^k(t) = \begin{cases} \frac{Q}{L^k(t)} & si\ (i,j) \in T^k(t) \\ 0 & si\ (i,j) \notin T^k(t) \end{cases} \tag{1.3}$$

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 :

$$\tau_{ij}(t+1) = (1-\rho) \cdot \tau_{ij}(t) + \Delta\tau_{ij}(t) \tag{1.4}$$

où $\Delta\tau_{ij}(t) = \sum_{k=1}^m \Delta\tau_{ij}^k(t)$ et m est le nombre de fourmis.

Algorithme 6 Algorithme à Estimation de Distribution

Engendrer aléatoirement M individus pour constituer la population D_0
Tant que le critère d'arrêt n'est pas atteint **faire**
$i = i + 1$
Sélectionner N individus dans D_{i-1} à l'aide d'un opérateur de sélection pour constituer la population D_{i-1}^{se}
Estimer la distribution de probabilité $pl(x) = p\left(x|D_{i-1}^{se}\right)$
Échantillonner M individus selon $pl(x)$ pour constituer D_i
Fin Tant que

Depuis, la démarche initiée par cette analogie a été étendue à la résolution d'autres problèmes d'optimisation, discrets et continus [Dorigo et al., 2005 ; Siarry et al., 2006]. Les algorithmes de colonies de fourmis présentent plusieurs caractéristiques intéressantes, telles que le parallélisme intrinsèque élevé, la robustesse (une colonie peut maintenir une recherche efficace même si certains de ses individus sont défaillants) ou encore la décentralisation (les fourmis n'obéissent pas à une autorité centralisée). De ce fait, ces algorithmes semblent être idéalement adaptés aux problèmes dynamiques, pour lesquels seule une information locale est disponible [Tfaili, 2007].

1.2.2.5 Algorithme à estimation de distribution

Les algorithmes à estimation de distribution (EDA pour *Estimation of Distribution Algorithm*) ont été conçus comme une variante des algorithmes évolutionnaires [Mülhenbein et al., 1996]. Cependant, ces méthodes n'utilisent ni opérateur de croisement, ni opérateur de mutation. Un modèle probabiliste est utilisé pour engendrer des solutions candidates, la distribution de probabilité étant estimée à chaque itération à partir des informations apportées par la précédente génération. L'estimation de la distribution de probabilité permet d'estimer les relations entre les différents individus, alors que celles-ci sont exprimées de manière implicite dans le cas des algorithmes évolutionnaires.

Le principe général d'un EDA est similaire à un algorithme évolutionnaire. On initialise tout d'abord aléatoirement une population D_0. Ensuite, à l'itération i, on utilise un opérateur de sélection pour déterminer la population D_{i-1}^{se}. L'information apportée par cette population est alors utilisée pour estimer la distribution de probabilité $pl(x)$ de l'itération courante. Enfin, la nouvelle population D_i est créée aléatoirement à partir de la distribution de probabilité $pl(x)$. Le parallèle avec les algorithmes évolutionnaires est ici respecté, le couple croisement-mutation étant remplacé par l'estimation de la distribution. La phase de diversification est ainsi assurée par l'utilisation d'une distribution de probabilité explicite, alors que la phase d'intensification n'est assurée que par la présence d'un opérateur de sélection.

Cette procédure est résumée par l'Algorithme 6.

La principale difficulté de cette méthode est le choix et l'estimation de la distribution de probabilité. En effet, il faut choisir au préalable quel modèle de distribution va être utilisé par l'algorithme et, à chaque itération, estimer les paramètres de cette distribution. De nombreux modèles de distributions ont été proposés dans la littérature [Larrañaga et al., 2001]. Ces modèles peuvent se diviser en trois catégories :

- Les modèles *sans dépendance* : la distribution de probabilité est factorisée à partir de distributions indépendantes univariantes pour chaque dimension. Ce cas a le défaut d'être le plus souvent inapplicable dans les cas réels, les variables étant la plupart du temps corrélées entre elles ;

- Les modèles à *dépendances bivariantes* : la distribution de probabilité est factorisée à partir de distributions bivariantes ;
- Les modèles à *dépendances multiples* : la distribution de probabilité est factorisée à partir de statistiques d'ordre supérieur à deux.

Il est à noter que, dans le cas continu, le modèle de distribution le plus souvent utilisé est la distribution gaussienne. L'utilisation de modèles avec dépendances introduit l'utilisation de réseaux probabilistes, les réseaux bayésiens ou gaussiens étant le plus souvent utilisés. Le nombre de modèles et le nombre d'algorithmes étant relativement élevés, il serait trop long de les exposer dans le présent manuscrit. On préférera se reporter au livre de référence dans le domaine [Larrañaga et al., 2001].

1.2.2.6 Optimisation par Essaim Particulaire

L'Optimisation par Essaim Particulaire (OEP) a été proposée par Kennedy et Eberhart en 1995 [Kennedy et al., 1995]. Cette méthode est inspirée du comportement social des animaux évoluant en essaim. L'exemple le plus souvent utilisé est le comportement des bancs de poissons [Wilson, 1975 ; Reynolds, 1987]. En effet, on peut observer chez ces animaux des dynamiques de déplacement relativement complexes, alors qu'individuellement chaque individu a une intelligence limitée et une connaissance seulement locale de sa situation dans l'essaim. Un individu de l'essaim n'a pour connaissance que la position et la vitesse de ses plus proches voisins. Chaque individu utilise donc, 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 de l'essaim, et qui permettent la mise en oeuvre de comportements collectifs complexes et adaptatifs. L' "intelligence globale" de l'essaim est donc la conséquence directe des interactions locales entre les différentes particules de l'essaim. La performance du système entier est supérieure à la somme des performances de ses parties.

Kennedy et Eberhart se sont inspirés de ces comportements socio-psychologiques pour créer l'OEP. Un essaim de particules, qui sont des solutions potentielles au problème d'optimisation, "survole" l'espace de recherche, en quête de 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 1.4 illustre la stratégie de déplacement d'une particule.

Dans un espace de recherche de dimension D, la particule i de l'essaim est modélisée par son vecteur position $\overrightarrow{X_i} = (x_{i1}, x_{i2}, \ldots, x_{iD})$ et par son vecteur vitesse $\overrightarrow{V_i} = (v_{i1}, v_{i2}, \ldots, v_{iD})$. Cette particule garde en mémoire la meilleure position par laquelle elle est déjà passée, que l'on note $\overrightarrow{p_i} = (p_{i1}, p_{i2}, \ldots, p_{iD})$. La meilleure position atteinte par toutes les particules de l'essaim est notée $\overrightarrow{g} = (g_1, g_2, \ldots, g_D)$. Au temps t, le vecteur vitesse est calculé à partir de (1.5).

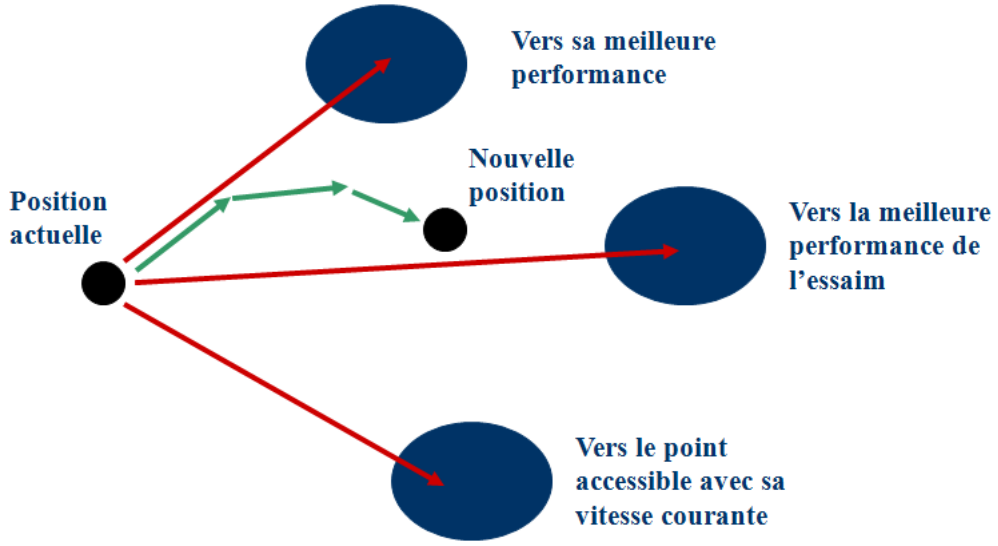


FIGURE 1.4 – Déplacement d'une particule.

$$v_{ij}(t) = w \cdot v_{ij}(t-1) + c_1 \cdot r_1 \cdot (p_{ij}(t-1) - x_{ij}(t-1)) + c_2 \cdot r_2 \cdot (g_j(t-1) - x_{ij}(t-1)),\ j \in \{1, \ldots, D\} \quad (1.5)$$

où w est en général une constante appelée, *coefficient d'inertie*, c_1 et c_2 sont deux constantes, appelées *coefficients d'accélération*, r_1 et r_2 sont deux nombres aléatoires tirés uniformément dans [0,1] à chaque itération et pour chaque dimension.

$w \cdot v_{ij}(t-1)$ correspond à la composante *physique* du déplacement. Le paramètre w contrôle l'influence de la direction de déplacement sur le déplacement futur . Il est à noter que, dans certaines applications, le paramètre w peut être variable [Siarry et al., 2006].

$c_1 \cdot r_1 \cdot (p_{ij}(t-1) - x_{ij}(t-1))$ correspond à la composante *cognitive* du déplacement. c_1 contrôle le comportement cognitif de la particule.

$c_2 \cdot r_2 \cdot (g_j(t-1) - x_{ij}(t-1))$ correspond à la composante *sociale* du déplacement. c_2 contrôle l'aptitude sociale de la particule.

La combinaison des paramètres w, c_1 et c_2 permet de régler la balance entre les phases diversification et intensification du processus de recherche [Shi et al., 1998 ; Kennedy et al., 2001]. Il est à noter que le terme "vitesse" est ici abusif car les vecteurs $\overrightarrow{V_i}$ ne sont pas homogènes à une vitesse. Il serait plus approprié de parler de "direction de déplacement". Cependant, pour respecter l'analogie avec le monde animalier, les auteurs ont préféré utiliser le terme "vitesse".

La position au temps t de la particule i est alors définie par (1.6).

$$x_{ij}(t) = x_{ij}(t-1) + v_{ij}(t),\ j \in \{1, \ldots, D\} \quad (1.6)$$

L'OEP 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 (1.5) et (1.6). Une fois le déplacement des particules effectué, les nouvelles positions sont évaluées. Les $\overrightarrow{p_i}$ ainsi que $\overrightarrow{g}$ sont alors mis à jour. Cette procédure est résumée par l'Algorithme 7. N est le nombre de particules de l'essaim.

Le critère d'arrêt peut être différent suivant le problème posé. Si l'optimum global est connu a priori, on peut définir une "erreur acceptable" ε comme critère d'arrêt. Sinon, il est commun de fixer un nombre maximum d'évaluations de la fonction objectif ou un nombre maximum d'itérations

Algorithme 7 Algorithme d'optimisation par essaim particulaire

Initialisation aléatoire des positions et des vitesses de chaque particule
Pour chaque particule i, $\overrightarrow{p_i} = \overrightarrow{X_i}$
Tant que le critère d'arrêt n'est pas atteint **faire**
  Pour $i = 1$ à N **faire**
    Déplacement de la particule à l'aide de (1.5) et (1.6)
    Évaluation des positions
    Si $f\left(\overrightarrow{X_i}\right) < f\left(\overrightarrow{p_i}\right)$
      $\overrightarrow{p_i} = \overrightarrow{X_i}$
    Fin Si
    Si $f\left(\overrightarrow{p_i}\right) < f\left(\overrightarrow{g}\right)$
      $\overrightarrow{g} = \overrightarrow{p_i}$
  Fin Pour
Fin Tant que

comme critère d'arrêt. Cependant, au regard du problème posé et des exigences de l'utilisateur, d'autres critères d'arrêt peuvent être utilisés.

1.2.2.6.1 Confinement des particules 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, on peut introduire un nouveau paramètre V_{max}, qui va permettre de contrôler l' "explosion" du système [Eberhart et al., 1996]. Une étude sur le comportement de l'OEP suivant les valeurs de V_{max} est disponible dans [Fan et al., 2001].

De plus, une stratégie de confinement des particules peut être introduite. Une telle stratégie permet de ramener une particule sortie de l'espace de recherche à l'intérieur de celui-ci. Plusieurs méthodes peuvent être alors employées :

- La particule est laissée à l'extérieur de l'espace de recherche, mais on n'évalue pas sa fonction objectif. Ainsi, elle ne pourra pas attirer les autres particules en dehors de l'espace de recherche ;
- La particule est stoppée à la frontière et les composantes correspondantes de la vitesse sont annulées ;
- La particule "rebondit" sur la frontière. La particule est stoppée sur la frontière, mais les composantes correspondantes de la vitesse sont multipliées par un coefficient a tiré aléatoirement dans l'intervalle [-1,0].

Cependant, suivant les besoins, de nombreuses autres méthodes peuvent être utilisées.

1.2.2.6.2 Graphes d'influence Si l'on se réfère à la version basique de l'OEP résumée par l'Algorithme 7, la meilleure particule $\overrightarrow{g}$ est choisie à partir de la population entière. Le graphe d'information est donc complètement connecté. On est ici en présence d'une topologie statique d'information, car les liens d'information entre les particules sont définis une fois pour toutes.

De même, Eberhart et al. [Eberhart et al., 1995] ont défini une version dite "locale" de l'OEP, qui utilise aussi un graphe d'information statique. Cette version utilise un graphe d'information "circulaire". Les particules de l'essaim sont virtuellement disposées en cercle et numérotées séquentiellement à partir de 1 en parcourant le cercle. La particule n'est donc plus informée par toutes les particules

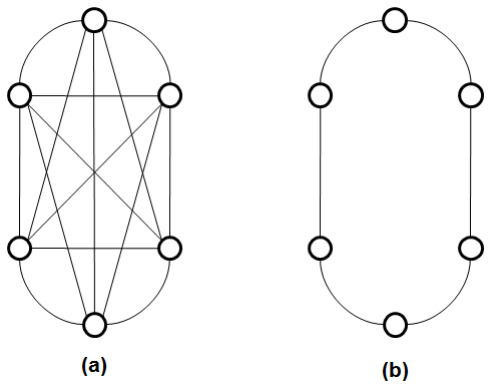


FIGURE 1.5 – Graphe d'influence d'un essaim de particules. (a) Graphe complètement connecté, (b) Graphe circulaire.

de l'essaim, mais par elle-même et ses deux voisins. Bien qu'elle converge moins rapidement que la version "globale", la version "locale" de l'OEP donne de meilleurs résultats, car elle est moins sujette à l'attraction par des minima locaux [Kennedy, 1999]. La Figure 1.5 illustre la différence entre un graphe complètement connecté et un graphe circulaire.

De nombreuses autres topologies ont été testées. Parmi celles-ci, on peut retenir celles inspirées de modèles socio-psychologiques [Bavelas, 1950] ou celles incluant des stratégies "petit-monde" [Watts et al., 1998]. Des résultats d'OEP utilisant différentes topologies sont disponibles dans [Kennedy, 1999 ; Clerc, 2006]. L'information importante qui ressort de ces résultats est qu'aucune topologie n'est "meilleure" que les autres, dans le sens où il n'y a aucune topologie qui donne de meilleurs résultats que toutes les autres sur un large panel de problèmes.

A l'inverse des topologies statiques, il existe aussi des topologies dynamiques. Dans ce cas, les liens d'information entre les particules sont modifiés à chaque pas de temps. Par exemple, dans [Suganthan, 1999], une stratégie "globale", favorisant une convergence rapide, est combinée à une stratégie "locale", qui favorise la phase de diversification. L'algorithme débute avec une topologie circulaire classique (2-voisinage) puis, au fur et à mesure du traitement, la taille des voisinages est augmentée jusqu'à obtenir un graphe complètement connecté. L'idée est de combiner les avantages des deux stratégies pour améliorer les résultats de l'OEP. De même, d'autres exemples de topologies aléatoires sont disponibles dans [Peram et al., 2003 ; Liang et al., 2005 ; Janson et al., 2005 ; Clerc, 2006]. Comme dans le cas des topologies statiques, les résultats ne permettent pas de dégager une hiérarchie claire.

1.2.2.6.3 Facteur de constriction L'étude de la dynamique des particules au sein de l'essaim a conduit à la recherche de solutions pour éviter la divergence de l'algorithme. Dans les paragraphes précédents, on a vu en quoi l'introduction du paramètre V_{max} peut limiter la divergence des particules. De nombreuses études sur la dynamique ont été publiées au fil du temps [Kennedy, 1998 ; Ozcan et al., 1999 ; Van den Bergh, 2002]. Toutes ces études s'attachent à étudier dans quelle situation une convergence de l'essaim est assurée.

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 ϕ permet de prévenir l'explosion de l'essaim, d'assurer la convergence, mais aussi de s'affranchir de la définition arbitraire d'un paramètre V_{max}. L'équation (1.5) devient alors :

$$v_{i,j}(t+1) = \chi\left(v_{i,j}(t) + \phi_1 \cdot r_1 \cdot (p_{i,j} - x_{i,j}(t)) + \phi_2 \cdot r_2 \cdot (g_j - x_{i,j}(t))\right),\ j \in \{1, \ldots, D\} \quad (1.7)$$

avec :

$$\chi = \frac{2}{\phi - 2 + \sqrt{\phi^2 - 4\phi}},\ \phi = \phi_1 + \phi_2,\ \phi > 4 \quad (1.8)$$

On remarque que l'OEP avec coefficient de constriction est équivalente à l'OEP originale avec $\chi \leftrightarrow w$, $c_1 \leftrightarrow \chi\phi_1$ et $c_2 \leftrightarrow \chi\phi_2$.

Dans [Clerc et al., 2002], de nombreux tests sont menés pour déterminer les valeurs optimales de ϕ_1 et ϕ_2. Dans la majorité des cas, on utilise $\phi = 4.1$ et $\phi_1 = \phi_2$; ce qui donne un coefficient multiplicatif χ approximativement égal à 0.7298.

La définition du paramètre ϕ permet de contrôler le système, afin qu'il présente les caractéristiques suivantes :

- Le système converge vers un état d'équilibre ;
- Plusieurs régions de l'espace de recherche sont explorées, tout en évitant une convergence prématurée.

Bien que le système converge avec certitude vers un état d'équilibre, il n'est cependant pas certain que l'algorithme converge vers l'optimum global. Il est à noter que l'OEP avec facteur de constriction n'est pas le seul algorithme d'OEP garantissant la convergence vers un état d'équilibre. D'autres exemples peuvent être trouvés dans [Van den Bergh, 2002; Trelea, 2003; Peer et al., 2003; Zheng et al., 2003].

La définition du facteur de constriction permet au système de converger, sans pour autant avoir à définir un paramètre V_{max}. Cependant, il a été montré dans [Eberhart et al., 2000], qu'il est quand même bénéfique de définir un paramètre V_{max}. Cette étude amène à définir, pour chaque dimension i, $V_{max,i}$ égal à $X_{max,i}$, qui définit la taille de l'espace de recherche sur la dimension i. Cette version de l'OEP a été adoptée comme la version canonique de celui-ci.

1.2.2.6.4 OEP et hybridation Procédé répandu dans le monde des métaheuristiques, l'hybridation a aussi touché le domaine de l'OEP. L'hybridation consiste à combiner les caractéristiques de deux méthodes différentes pour tirer les avantages des deux méthodes [Talbi, 2002]. Dans ce paragraphe, on présentera quelques exemples d'hybridation entre l'OEP et d'autres méthodes.

La première hybridation d'un algorithme d'OEP est rencontrée dans [Angeline, 1998]. Dans cet article, Angeline introduit dans l'algorithme d'OEP un processus de sélection et un processus de mutation inspirés des algorithmes évolutionnaires. Le processus de sélection est utilisé pour choisir des "bonnes" particules qui vont subir une mutation et des "mauvaises " particules qui sont éliminées. Dans [Miranda et al., 2002], Miranda et al. utilisent aussi une hybridation entre l'OEP et les stratégies évolutionnaires. Les paramètres χ, ϕ_1 et ϕ_2, ainsi que $\overrightarrow{g}$, sont perturbés à l'aide d'une distribution gaussienne. La variance de cette distribution est déterminée à l'aide d'un processus de sélection. Dans [Robinson et al., 2002], il est développé une hybridation entre l'OEP et les algorithmes génétiques. Il est montré dans cet article que l'OEP est favorable dans la phase de diversification alors que les algorithmes génétiques sont plus efficaces dans la phase d'intensification. Dans [Poli et al., 2004], on trouve une hybridation entre l'OEP et la méthode du "hill-climbing" [Russell et al., 2003]. Les particules n'ont plus ici de mémoire, mais utilisent plutôt la physique des masses pour se déplacer dans l'espace de recherche. Des forces telles que la gravité, l'élasticité ou le frottement sont notamment utilisées. La gravité apporte la compétence permettant de chercher les mimina,

l'élasticité permet de maintenir la diversité et les frottements permettent d'intensifier la recherche dans les zones intéressantes de l'espace de recherche. Dans [Shelokar et al., 2004], un algorithme d'OEP est hybridé avec un algorithme de colonies de fourmis. L'OEP est utilisée comme méthode de recherche globale alors que l'algorithme de colonies de fourmis est censé améliorer le processus d'intensification, en étant utilisé comme une recherche locale. Une hybridation entre un algorithme d'OEP et un procédé d'évolution différentielle [Price, 1996] est présentée dans [Zhang et al., 2003]. Enfin, un algorithme d'OEP utilisant des principes des algorithmes à estimation de distribution est présenté dans [Iqbal et al., 2006]. Les meilleures particules sont ici utilisées pour attirer les autres particules de l'essaim à l'aide de l'estimation d'une distribution de probabilité.

1.2.2.6.5 OEP adaptative Comme toutes les autres métaheuristiques, l'OEP possède de nombreux paramètres de contrôle. Si l'on se réfère à l'OEP avec facteur de constriction, les paramètres à définir par l'utilisateur sont les suivants :

- N, la taille de l'essaim ;
- K, la taille maximale du voisinage d'une particule ;
- ϕ, le coefficient de constriction ;
- ϕ_1 et ϕ_2 ;
- V_{max}, la vitesse maximale.

Le réglage de tous ces paramètres est un processus qui s'avère long et difficile. En effet, chaque paramètre ayant une forte influence sur le comportement de l'algorithme [Shi et al., 1998 ; Van den Bergh, 2002], il est important de trouver un jeu de paramètres bien adapté au problème posé. De ce fait, chaque problème nécessiterait en théorie une étude qui permettrait de dégager le jeu de paramètres optimal pour le traiter. Or, un tel procédé est souvent long à réaliser et demande une bonne connaissance au préalable du comportement de l'algorithme, ce qui le rend inapplicable dans un contexte industriel. C'est pourquoi un nouvel axe de recherche, qui s'attache à réduire le nombre de paramètres "libres" des métaheuristiques, s'est développé. Le but est de trouver des règles qui affranchissent l'utilisateur de définir les paramètres, en calculant ceux-ci au fur et à mesure du traitement, en fonction des résultats donnés par l'algorithme. Ainsi, l'algorithme "adapte" son comportement au problème posé, au lieu d'avoir un comportement figé, défini au préalable par l'utilisateur. Le but ultime serait de concevoir un algorithme qui n'aurait aucun paramètre de contrôle ; un tel algorithme agirait comme une "boîte noire" pour laquelle l'utilisateur n'aurait qu'à définir les problèmes et le critère d'arrêt. Évidemment, pour être valable, un tel algorithme se doit de présenter des résultats au moins égaux à ceux des algorithmes paramétriques.

Dans le but de limiter l'intervention de l'utilisateur, des études sur le choix des paramètres ont été mises en place au cours des dernières années. Dans [Shi et al., 1998], Shi et Eberhart proposent une étude qui donne une indication sur les performances de l'OEP en fonction des différentes valeurs de w et de V_{max}. Dans [Van den Bergh, 2002], Van den Bergh propose une étude exhaustive de l'influence des paramètres sur le comportement de trois algorithmes différents d'OEP. Il y apparaît que, suivant que le problème posé soit unimodal ou multimodal, les jeux de paramètres sont différents. L'enjeu crucial apparaît être le taux de diversité au sein de l'essaim, un problème multimodal nécessitant un essaim avec un taux de diversité beaucoup plus important que dans le cas unimodal. Une topologie de voisinage fortement connectée favorisera ainsi la résolution des problèmes unimodaux, alors qu'un essaim de taille importante aura tendance à favoriser la résolution de problèmes multimodaux. Dans [Trelea, 2003], l'OEP est étudiée en utilisant les résultats de base sur les systèmes dynamiques. Comme Van den Bergh, Trelea pose la définition des paramètres comme un problème de compromis diversification-intensification. De nombreuses autres études sont disponibles

dans la littérature. Cependant, bien que donnant une idée aux utilisateurs potentiels de l'influence des paramètres, ces études sont quand même relativement restrictives. En effet, ces études portent souvent sur des intervalles limités de valeurs et ne sont applicables qu'à des topologies statiques de voisinages.

Il existe aussi des études portant sur la définition des procédures automatiques de choix des paramètres. Birattari et al. [Birattari et al., 2002] proposent une procédure de configuration des métaheuristiques basée sur un algorithme nommé F-RACE, qui appartient à la classe des "racing algorithms" très utilisés en apprentissage [Maron et al., 1994]. Cet algorithme utilise le test non-paramétrique de Friedman [Friedman, 1937] pour adapter au mieux le jeu de paramètres aux spécificités du problème posé. Hutter et al. [Hutter et al., 2006] proposent d'utiliser des modèles "hardness" pour pouvoir prédire les paramètres d'un algorithme. Adenso-Diaz et Laguna [Adenso-Diaz et al., 2006] proposent eux-aussi une méthode nommée CALIBRA, qui utilise la méthode de Taguchi [Nair, 1992], ainsi qu'un algorithme de recherche locale, pour déterminer les paramètres. Ces procédures, bien qu'efficaces sur certains points, n'ont pour autant pas rencontré un vif succès. La principale raison est qu'elles ne s'appliquent qu'à un champ restreint de méthodes et elles ne se sont pas avérées robustes sur un large panel de problèmes.

Enfin, la dernière façon de choisir les paramètres d'une méthode est justement de ne pas les choisir. Au lieu de définir des paramètres avant l'exécution de la méthode, on préfère ici modifier les valeurs des paramètres au cours du traitement, en fonction des résultats trouvés au cours des différentes itérations. De nombreuses métaheuristiques ont été conçues dans ce domaine : des algorithmes génétiques [Schnecke et al., 1996; Sawai et al., 1999; Murata et al., 2002], des algorithmes de colonies de fourmis [Chen et al., 2004; Di Caro, 1998b; Förster et al., 2007], des algorithmes de recherche tabou [Battiti, 1996] ou encore des algorithmes de recuit simulé [Ingber, 1996].

Dans le domaine de l'OEP, de nombreux algorithmes de ce type existent aussi. Suganthan [Suganthan, 1999] propose de modifier la taille des voisinages des particules en fonction du temps. De même, Shi [Shi et al., 2001] propose de modifier le coefficient d'inertie w en utilisant des règles de logique floue. L'algorithme proposé par Ye et al. [Ye et al., 2002] élimine les particules "inactives" et les remplace par de nouvelles particules, plus à même d'explorer l'espace de recherche. Zhang et al. [Zhang et al., 2003b] proposent de modifier à la fois la taille de l'essaim, le coefficient de constriction et la taille des voisinages, à l'aide d'un seuil d' "amélioration". Dans [Kennedy, 2003], Kennedy propose de remplacer les équations de déplacement par une distribution gaussienne de moyenne $\frac{\overrightarrow{p_i}+\overrightarrow{p_g}}{2}$ et de variance $|\overrightarrow{p_i} - \overrightarrow{p_g}|$. En 2004, Yasuda et Iwasaki [Yasuda et al., 2004] ont proposé un algorithme qui définit les valeurs des paramètres à partir de l'information apportée par les vecteurs vitesse des particules. En 2003, Clerc [Clerc, 2003] a développé un algorithme adaptatif appelé TRIBES. La description de TRIBES fera l'objet du Chapitre 2. Cette liste est non exhaustive, mais dresse un aperçu assez complet de tout ce qui peut être imaginé pour adapter un algorithme d'OEP.

Ce paragraphe a présenté succinctement l'Optimisation par Essaim Particulaire, ainsi que les principales avancées depuis sa création en 1995. Pour plus de détails sur la méthode, une revue complète est disponible dans [Banks et al., 2007 ; Banks et al., 2007b].

1.3 Optimisation multiobjectif

Les métaheuristiques ont toutes été conçues à l'origine pour résoudre des problèmes mono-objectif. Cependant, devant l'omniprésence des problèmes multiobjectifs dans les cas réels, de nouvelles méthodes ont dû être créées pour résoudre de tels problèmes. Après avoir défini le problème d'optimisation multiobjectif, cette partie présentera de manière non exhaustive un état de l'art des

métaheuristiques conçues pour résoudre les problèmes multiobjectifs.

1.3.1 Définition du problème

Un problème d'optimisation multiobjectif est un problème du type :

$$Minimiser\ \overrightarrow{f}(\overrightarrow{x}) = (f_1(\overrightarrow{x}),\ f_2(\overrightarrow{x}),\ldots,\ f_k(\overrightarrow{x})) \tag{1.9}$$

sous les contraintes suivantes :

$$g_i(\overrightarrow{x}) \leq 0,\ i = 1,\ 2,\ldots,m \tag{1.10}$$

$$h_i(\overrightarrow{x}) = 0,\ i = 1,\ 2,\ldots,p \tag{1.11}$$

où $\overrightarrow{x} = (x_1,\ x_2,\ldots,\ x_D)$ est une position dans l'espace de recherche, $f_i : \mathbb{R}^D \rightarrow \mathbb{R}$, i=1,...,k, sont les fonctions objectifs du problème, $g_i : \mathbb{R}^D \rightarrow \mathbb{R}$, i=1,...,m, sont les contraintes d'inégalité du problème et $h_i : \mathbb{R}^D \rightarrow \mathbb{R}$, i=1,...,p, sont les contraintes d'égalité du problème.

Le problème de l'optimisation multiobjectif réside dans le fait que que l'on ne sait pas "classer" suivant plusieurs critères car $\mathbb{R}^k$ ne possède pas de relation d'ordre total si $k \neq 1$.

Dans le cas multiobjectif, le concept d'optimum est différent que dans le cas mono-objectif. En effet, on n'est plus ici à la recherche d'un unique optimum global, mais plutôt d'une surface de solutions qui offrent un bon "compromis" entre les différents objectifs. Pour bien comprendre la notion d'optimalité dans le cas multiobjectif, on introduit les définitions suivantes, basées sur les travaux de Pareto [Pareto, 1896] :

- **Définition 1** : Étant donné $\overrightarrow{x},\ \overrightarrow{y} \in \mathbb{R}^k$, on dit que $\overrightarrow{x} \leq \overrightarrow{y}$ si $x_i \leq y_i$ pour i=1,..., k, et que $\overrightarrow{x}$ **domine** $\overrightarrow{y}$ (noté $\overrightarrow{x} \prec \overrightarrow{y}$) si $\overrightarrow{x} \leq \overrightarrow{y}$ et $\overrightarrow{x} \neq \overrightarrow{y}$.

La Figure 1.6 illustre un cas de dominance dans le cas d'une minimisation de deux fonctions objectif.

- **Définition 2** : On dit qu'un vecteur solution $\overrightarrow{x} \in \chi \subset \mathbb{R}^D$ est **non-dominé** dans χ s'il n'existe pas d'autre vecteur $\overrightarrow{x'} \in \chi$ tel que $\overrightarrow{f}\left(\overrightarrow{x'}\right) \prec \overrightarrow{f}(\overrightarrow{x})$.
- **Définition 3** : On dit qu'un vecteur solution $\overrightarrow{x^*} \in \mathcal{F} \subset \mathbb{R}^D$ est **Pareto-optimal** s'il est non-dominé dans $\mathcal{F}$.
- **Définition 4** : L'**ensemble optimal de Pareto** est défini par :
$$P^* = \{\overrightarrow{x} \in \mathcal{F} |\ x\ est\ Pareto - optimal\} \tag{1.12}$$
- **Définition 5** : Le **front optimal de Pareto** est défini par :
$$\mathcal{F}_{P^*} = \left\{\overrightarrow{f}(\overrightarrow{x}) \in \mathbb{R}^k |\ \overrightarrow{x} \in P^*\right\} \tag{1.13}$$

La Figure 1.7 montre le cas particulier d'un front de Pareto dans le cas d'un problème de minimisation bi-objectif.

Le but d'un algorithme d'optimisation multiobjectif est donc de trouver les positions de l'espace de recherche qui correspondent au front de Pareto $\mathcal{F}_{P^*}$. Les algorithmes d'optimisation multiobjectif se divisent en trois familles : les approches agrégatives, les approches non-Pareto et les approches Pareto.

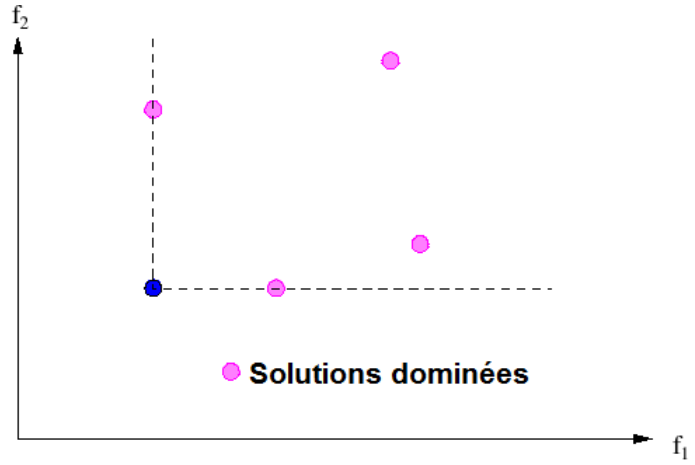


FIGURE 1.6 – Relation de dominance.

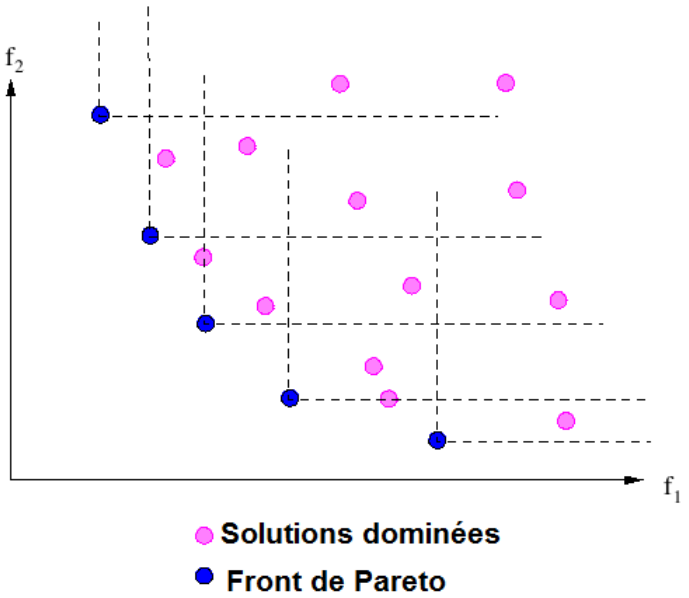


FIGURE 1.7 – Front de Pareto dans le cas bi-objectif.

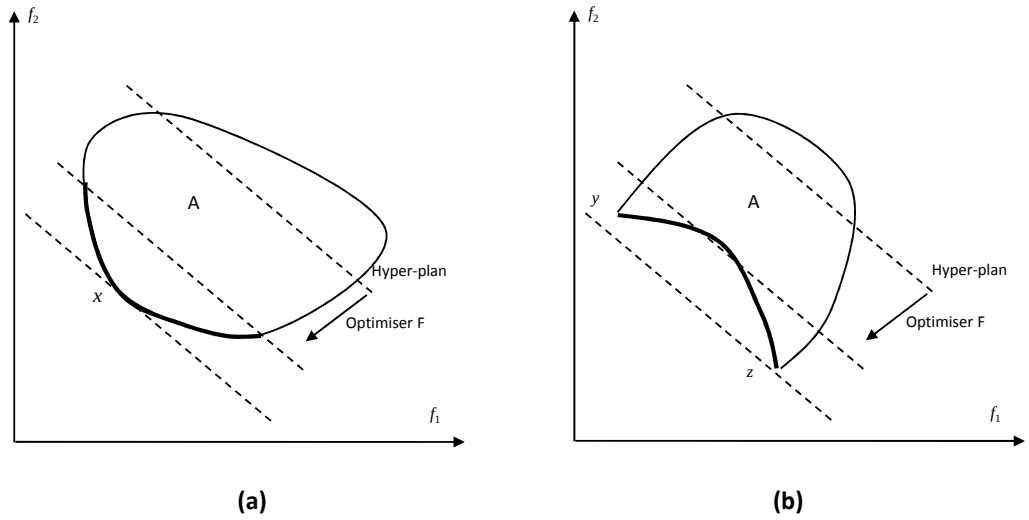


FIGURE 1.8 – Méthode d'agrégation. (a) Front de Pareto convexe, (b) Front de Pareto concave.

1.3.2 Approche agrégative

Cette approche, dite approche naïve, consiste à transformer un problème multiobjectif en problème mono-objectif pour y appliquer les méthodes déjà existantes [Collette et al., 2002]. Pour cela, on réalise une combinaison des f_i pour obtenir un objectif unique F.

$$F(\overrightarrow{x}) = \sum_{i=1}^{k} a_i f_i(\overrightarrow{x}) = \left\langle \overrightarrow{a} \mid \overrightarrow{f} \right\rangle \tag{1.14}$$

où les paramètres a_i sont strictement positifs et $\sum_{i=1}^{k} a_i = 1$. Ils représentent les poids affectés aux différents critères. D'un point de vue géométrique, l'agrégation peut être vue comme la projection du vecteur objectif $\overrightarrow{f}$ sur le vecteur "poids" $\overrightarrow{a}$.

Cette méthode présente pour avantage d'être simple à mettre en oeuvre et elle ne fournit qu'une seule solution, ce qui évite l'intervention d'un décideur. Les principaux problèmes sont la détermination des paramètres a_i et le choix des interactions entre les différents critères.

Il est commun de ne pas donner aux a_i des valeurs statiques, mais plutôt de modifier leurs valeurs au cours du temps. Par exemple, on peut trouver dans la littérature des méthodes "aveugles", qui permettent de modifier de façon aléatoire les paramètres au cours du temps [Ishibuchi et al., 1998].

La Figure 1.8 (a) illustre le fonctionnement de la méthode d'agrégation dans le cas d'une minimisation bi-objectif. L'espace des solutions réalisables est noté A. Le calcul des paramètres revient à trouver un hyperplan tangent au front de Pareto. La Figure 1.8 (b) illustre les limites de la méthode. Si le front de Pareto est concave, seules solutions y et z peuvent être trouvées par la recherche d'un plan tangent au front de Pareto, les autres points du front de Pareto ne pouvant être "tangentés". Cette méthode est donc limitée car le choix préalable d'une direction de projection introduit une perte d'information.

1.3.3 Approche non-Pareto

Les méthodes basées sur une approche non-Pareto ont pour caractéristique de traiter les objectifs séparément. Deux groupes de méthodes existent : les méthodes à sélection lexicographique et les

méthodes à sélection parallèle.

Dans une approche classique de sélection lexicographique, les fonctions sont optimisées séquentiellement, suivant un ordre défini a priori. Cet ordre permet de définir le poids des objectifs. Plusieurs métaheuristiques ont été utilisées pour la résolution des problèmes multiobjectifs à sélection lexicographique [Talbi, 1999].

L'approche à sélection parallèle a été proposée par Schaffer dans [Schaffer, 1985]. Son algorithme, inspiré d'un algorithme évolutionnaire et nommé VEGA, sélectionne les solutions courantes du front de Pareto suivant chaque objectif, indépendamment des autres (d'où le parallélisme). L'analyse du comportement de cet algorithme a montré qu'il se comportait d'une manière similaire à un algorithme utilisant une méthode agrégative. D'autres travaux ont été menés sur ce sujet [Richardson et al., 1989 ; Berro, 2001].

1.3.4 Approche Pareto

Contrairement aux deux précédentes approches, l'approche Pareto utilise la notion de dominance pour sélectionner des solutions faisant converger la population vers un ensemble de solutions qui approchent avec justesse le front de Pareto. Cette approche a été initiée par Goldberg en 1989 [Goldberg, 1989]. Cette approche assure un traitement équitable de chaque critère, car il n'y a pas de classement a priori de l' "importance" des critères. Ainsi, en fin de traitement, l'algorithme fournit un ensemble de solutions qui approchent le front de Pareto. Le choix de la solution finale revient donc à l'utilisateur, qui doit choisir parmi l'ensemble fourni la solution qui lui convient le mieux [Talbi, 1999]. Ces méthodes se sont avérées être les plus efficaces donc, de nos jours, la majorité des algorithmes utilisent une approche Pareto pour traiter les problèmes multiobjectifs. Le paragraphe suivant fournit un bref aperçu des métaheuristiques d'optimisation multiobjectif.

1.3.5 Métaheuristiques pour l'optimisation multiobjectif

1.3.5.1 Algorithme de recuit simulé

La méthode de recuit simulé en optimisation multiobjectif a d'abord été abordée sous l'angle agrégatif [Serafini, 1992 ; Friesz et al., 1993]. Les deux méthodes les plus populaires sont la méthode MOSA ("Multiple Objective Simulated Annealing") proposée par Ulungu et al. [Ulungu et al., 1999] et la méthode PASA ("Pareto Archived Simulated Annealing") proposée dans [Engrand, 1997]. MOSA utilise les caractéristiques du recuit simulé pour rechercher le plus efficacement possible les solutions non-dominées. PASA utilise une fonction d'agrégation des fonctions objectifs, couplée avec un système d'archivage des solutions non-dominées.

1.3.5.2 Algorithmes évolutionnaires

Les algorithmes évolutionnaires sont très largement utilisés pour résoudre des problèmes multiobjectifs. Parmi tous les algorithmes existants, on retrouve les trois approches décrites en 1.3.2, 1.3.3 et 1.3.4 [Talbi, 1999]. Une étude comparative des algorithmes évolutionnaires pour l'optimisation multiobjectif est disponible dans [Zitzler et al., 1999]. Parmi les méthodes utilisant l'approche Pareto, on distingue deux familles d'algorithmes : les non-élitistes et les élitistes.

Parmi les méthodes non-élitistes, l'algorithme MOGA ("Multiple Objective Genetic Algorithm") proposé par Fonseca et al. [Fonseca et al., 1995] est l'un des plus connus. Cet algorithme se base sur le classement des individus suivant le nombre d'individus dominés. NSGA ("Non Dominated Sorting Genetic Algorithm") [Srivinas et al., 1995] propose de diviser la population en plusieurs groupes,

Algorithme 8 Algorithme général d'OEP multiobjectif

Initialisation aléatoire des positions et des vitesses de chaque particule
Déterminer *les leaders*
Stockage des solutions non-dominées
Tant que le critère d'arrêt n'est pas atteint **faire**
  Pour $i = 1$ à N **faire**
    Déplacement de la particule à l'aide de (1.5) et (1.6)
    Mutation
    Évaluation des positions
    Mise à jour *des leaders cognitifs*
  Fin Pour
  Mise à jour *des leaders sociaux*
  Mise à jour *des solutions non-dominées*
Fin Tant que

fonctions du degré de domination de chaque individu. Enfin, l'algorithme qui fait référence dans le domaine est NSGA-II [Deb et al., 2002], qui est une évolution de NSGA, mais en version élitiste.

Parmi les méthodes élitistes, on peut citer SPEA ("Stength Pareto Evolutionary Algorithm") [Ziztler et al., 1999], où l'on profite du passage de génération pour mettre à jour les différentes sauvegardes. Les individus non-dominés sont sauvegardés et les individus dominés déjà présents sont éliminés. Une évolution de cet algorithme est disponible dans [Zitzler et al., 2002]. Cette évolution a une plus grande complexité, mais présente de meilleurs résultats.

1.3.5.3 Algorithme de colonies de fourmis

En optimisation mono-objectif, les algorithmes de colonies de fourmis sont très appréciés pour résoudre des problèmes combinatoires. En revanche, peu de travaux existent dans le cas multiobjectif. Dans l'état de l'art réalisé par Blum en 2005 [Blum, 2005] on ne retrouve que trois références à des algorithmes multiobjectifs. Doerner et al. [Doerner et al., 2004 ; Doerner et al., 2006] ont proposé un algorithme nommé P-ACO dédié à la résolution du problème d'allocation de portefeuilles. Cet algorithme présente de bons résultats, notamment face à NSGA et au recuit simulé, mais souffre de ne pas avoir été comparé à NSGA-II, la référence en la matière. L'algorithme OCF de Gagné [Gagné et al., 2004] repose sur le même principe que P-ACO. A chaque itération, les fourmis changent de fonction objectif à optimiser. A la fin de l'itération, la fourmi ayant la meilleure performance met à jour le tracé de phéromone. Quelques travaux sur une application de poursuite des multi-trajets sont disponibles dans [Benlian et al., 2007].

1.3.5.4 Optimisation par Essaim Particulaire

L'Algorithme 8 présente le schéma global d'un algorithme d'OEP adapté pour l'optimisation multiobjectif. Les étapes ajoutées par rapport à une OEP mono-objectif sont indiquées en italique.

La résolution d'un problème multiobjectif revient à sélectionner les positions non-dominées trouvées par l'algorithme au cours de son exécution. De ce fait, à la fin de l'exécution, les solutions retenues se doivent d'être des solutions non-dominées vis-à-vis de toutes les positions atteintes par les particules au cours des itérations successives. Il apparaît donc nécessaire de tenir à jour une liste des solutions non-dominées trouvées au cours du traitement et de mettre celle-ci à jour à la fin de chaque itération. La solution la plus couramment utilisée pour résoudre ce problème est d'utiliser une archive extérieure. Une solution potentielle est alors acceptée dans l'archive si, et seulement si,

elle est non-dominée vis-à-vis du contenu de l'archive. De plus, si elle domine des solutions contenues dans l'archive, celles-ci sont retirées de l'archive.

Le problème d'une telle approche est qu'elle peut être très rapidement coûteuse en complexité. En effet, la taille de l'archive peut devenir rapidement importante. De ce fait, le temps de calcul pour mettre à jour les solutions non-dominées peut s'avérer excessif. Dans le pire des cas, toutes les positions de l'essaim sont ajoutées à l'archive, ce qui induit une complexité de $\mathcal{O}\left(kMN^2\right)$ où k est le nombre d'objectifs, M est le nombre d'itérations et N est la taille de l'essaim. Pour pallier cet inconvénient, il est commun de définir une taille maximale pour l'archive [Coello Coello et al., 2002]. L'introduction de ce paramètre implique aussi la définition d'une règle d'inclusion de solutions dans l'archive, dans le cas où celle-ci est pleine.

Dans les paragraphes suivants, on dresse une liste non exhaustive des travaux existants en OEP multiobjectif.

1.3.5.4.1 Approche agrégative Parsopoulos et Vrahatis [Parsopoulos et al., 2002] proposent d'agréger les objectifs suivant trois méthodes différentes : une agrégation linéaire classique, pour laquelle les poids sont fixés, une agrégation dynamique, pour laquelle les poids sont modifiés au cours du traitement et une agrégation dite "bang-bang" [Jin et al., 2001], pour laquelle les poids sont brutalement modifiés au cours du temps. Cet algorithme utilise une OEP avec une topologie totalement connectée.

De même, Baumgartner et al. [Baumgartner et al., 2004] utilisent une OEP avec une topologie complètement connectée combinée avec une agrégation linéaire des objectifs. L'essaim est divisé en n sous-essaims utilisant chacun un jeu de poids différent et possédant chacun leur propre leader. Les solutions optimales au sens de Pareto sont déterminées à l'aide d'une méthode basée sur l'utilisation du gradient.

1.3.5.4.2 Approche non-Pareto Hu et Eberhart [Hu et al., 2002] ont proposé un algorithme qui optimise un seul objectif à la fois, en utilisant un ordre lexicographique. Cette approche utilise une OEP avec une topologie circulaire de voisinage. Cet algorithme n'introduit pas d'archive externe de stockage. Cependant, dans [Hu et al., 2003], une version de cet algorithme avec utilisation d'une archive externe est proposée.

Parsopoulos et al. [Parsopoulos et al., 2004] ont proposé une version parallèle de VEPSO ("Vector Evaluated Particle Swarm") inspirée de VEGA (cf. 1.3.5.2). VEPSO est un algorithme multi-essaim pour lequel chaque sous-essaim traite un des objectifs. Chaque sous-essaim échange son information (e.g., son $\overrightarrow{g}$) avec les autres sous-essaims. Les auteurs montrent que cette méthode peut mener à des solutions Pareto optimales.

Dans [Chow et al., 2004], un algorithme multi-essaim est aussi proposé. Chaque essaim traite individuellement un des objectifs. L'échange des informations est effectué entre les sous-essaims voisins par l'intermédiaire de l'échange des vecteurs $\overrightarrow{g}$. A l'intérieur d'un sous-essaim les particules sont complètement connectées. A l'opposé, une topologie locale de voisinage est utilisée pour connecter les différents sous-essaims.

1.3.5.4.3 Approche Pareto Dans [Ray et al., 2002], il est proposé un algorithme utilisant la dominance de Pareto et qui combine l'OEP avec des techniques issues des algorithmes évolutionnaires. Les auteurs utilisent un opérateur de densité sur le voisinage pour promouvoir la diversité au sein de l'essaim.

Coello Coello et al. [Coello Coello et al., 2002b ; Coello Coello et al., 2004] ont développé un algorithme basé sur l'utilisation d'une archive externe. La mise à jour de l'archive est réalisée suivant

un critère géographique. L'espace de recherche est divisé en hyper-cubes auxquels on donne des "notes", qui sont fonction du nombre de solutions non-dominées situées dans chacun d'eux. Les leaders sont choisis dans les hyper-cubes sélectionnés à l'aide d'un opérateur de sélection par "roulette" basé sur leurs notes. Un opérateur de mutation est aussi utilisé.

Une approche hybride OEP-algorithme évolutionnaire est proposée dans [Srinivasan et al., 2003]. Le but ici est d'appliquer les opérateurs des algorithmes évolutionnaires afin de rendre plus efficaces les mécanismes de l'OEP. Un opérateur de sélection est ici aussi utilisé pour assurer la convergence vers des solutions non-dominées. Les auteurs n'utilisent pas d'archive externe, la population à la dernière itération constituant les solutions finales du problème.

Bartz-Beielstein et al. [Bartz-Beielstein et al., 2003] ont introduit une stratégie élitiste à l'OEP. Différents opérateurs de sélection et d'élimination sont étudiés pour trouver la combinaison qui produit la meilleure approximation du front de Pareto. Les méthodes d'élimination sont basées sur la contribution des particules à la diversité de l'essaim. A l'opposé, les opérateurs de sélection sont basés sur les valeurs des fonctions objectifs.

Dans [Li, 2003], on trouve une adaptation des principaux mécanismes de NSGA-II à l'OEP multiobjectif. Les leaders sont sélectionnés parmi les éléments de l'archive. On utilise deux méthodes de sélection : une stratégie de "niching" et une stratégie de densité du voisinage. De même, dans [Raquel et al., 2005], la distance d'encombrement (*crowding distance)*, déjà utilisée dans NSGA-II, sert de critère pour maintenir la diversité dans l'archive. La sélection des leaders se fait aussi à partir des éléments de l'archive.

De nombreuses autres méthodes sont référencées dans [Reyes-Sierra et al., 2006].

1.4 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.

Un intérêt particulier a été porté à la méthode d'optimisation par essaim particulaire. Cette jeune méthode, inspirée des déplacements d'animaux en essaims, a rencontré un vif succès depuis sa création. Sa relative simplicité et son efficacité en font un des algorithmes les plus utilisés de nos jours. De nombreux axes de recherches ont pour objet de tirer le meilleur parti du paradigme de l'OEP. Parmi ceux-ci, la réduction des paramètres apparaît comme un enjeu crucial, en vue de l'introduction massive de l'OEP dans l'industrie. Les chapitres suivants présentent une méthode d'optimisation particulaire adaptative, qui présente la particularité de ne comporter aucun paramètre de contrôle.

Chapitre 2

Étude et perfectionnement de TRIBES, algorithme d'OEP sans paramètre

2.1 Introduction

Comme nous l'avons vu dans le Chapitre 1 de ce manuscrit, la famille des métaheuristiques regroupe un panel très varié de méthodes d'optimisation. Sujet privilégié dans le domaine de la recherche opérationnelle, elles arrivent aujourd'hui à un tournant important de leur existence. En effet, les différentes méthodes, telles qu'elles ont été initialement définies, atteignent aujourd'hui leurs limites. De ce fait, de nouveaux axes de recherche sont explorés afin de dépasser les restrictions imposées par les algorithmes originaux et, ainsi, d'arriver à une meilleure prise en main des algorithmes par les utilisateurs novices, mais aussi à de meilleures performances.

Les métaheuristiques sont reconnues pour être des méthodes performantes mais elles ne rencontrent qu'un succès modéré dans le monde de l'industrie. Dans un milieu où seule la performance compte, l'aspect stochastique des métaheuristiques semble encore être un obstacle difficile à franchir pour les décisionnaires. Il est donc important que les chercheurs de la communauté portent un effort tout particulier sur la facilité de prise en main des algorithmes. Plus les algorithmes seront faciles d'accès pour les utilisateurs novices, plus l'utilisation de ceux-ci pourra se répandre.

Parmi les améliorations possibles, la réduction du nombre de paramètres des algorithmes apparaît comme un enjeu majeur. En effet, les métaheuristiques sont fortement dépendantes de leur jeu de paramètres. De ce fait, le traitement d'un problème passe, au préalable, par la définition d'un jeu de paramètres optimal. Or, cette étape, déjà longue et fastidieuse, nécessite un connaissance assez poussée des mécanismes de l'algorithme utilisé. Un industriel ayant un problème d'optimisation à résoudre risque de ne pas prendre le temps de trouver le jeu de paramètres optimal. Il est donc important d'affranchir l'utilisateur de cette étape en proposant des règles "intelligentes" qui permettent de déterminer automatiquement les paramètres des algorithmes.

De telles règles ont déjà été présentées dans le cadre de l'optimisation par essaim particulaire (cf. section 1.2.2.6). Dans ce chapitre, on présente en détail TRIBES, un algorithme d'optimisation par essaim particulaire sans paramètres de contrôle [Clerc, 2003]. Les contributions apportées à cet algorithme seront aussi présentées, ainsi que des résultats numériques établis à partir de la procédure de test définie lors de la conférence *2005 IEEE Congress on Evolutionary Computation*.

2.2 Présentation de TRIBES

La description d'un problème d'optimisation impose à l'utilisateur de spécifier un certain nombre de données que l'algorithme ne peut pas "deviner". Tout d'abord, il doit délimiter l'espace de recherche, souvent en précisant pour chaque dimension l'intervalle des valeurs admissibles. Ensuite, il doit indiquer comment, en chaque point de l'espace, la fonction objectif est évaluée. Enfin, il doit préciser son critère d'arrêt. Une fois ces trois éléments définis, le problème est totalement décrit.

Définir un algorithme sans paramètres de contrôle implique qu'une fois le problème à résoudre spécifié, l'utilisateur n'a plus aucun paramètre à définir. L'algorithme doit donc incorporer des règles qui vont définir comment, à chaque itération de l'algorithme, l'essaim va évoluer et se comporter, tout en intégrant les informations progressivement recueillies au cours du traitement. Définir de telles règles peut apparaître comme une manière détournée de donner des instructions de fonctionnement au programme, donc de définir des paramètres. Cependant, si ces règles sont suffisamment robustes et générales, il est possible de les ignorer et de les considérer comme faisant partie de manière intégrante de la méthode. Par exemple, on pourrait coder en dur une règle du type "on utilise 50 particules" et dire que la définition de ce paramètre est évitée, mais l'expérience montre que les résultats sont mauvais pour certains problèmes et bons pour d'autres ; la règle n'est pas robuste.

Considérer ce point de vue revient à adopter un point de vue "ingénieur" plutôt qu'un point de vue "chercheur". Les résultats recherchés ne sont pas forcément les résultats optimaux, mais du moins une approximation "correcte" de l'optimum. Ce qui est perdu en efficacité est gagné en facilité de prise en main de l'algorithme. En effet, il est absolument normal qu'un algorithme qui doit rechercher par lui-même les valeurs de ses propres paramètres présente des résultats inférieurs à un algorithme dont le comportement est "guidé" par des paramètres soigneusement définis par l'utilisateur.

Clerc [Clerc, 2006] a adopté ce point de vue pour définir TRIBES, un algorithme d'OEP adaptatif. Au sens vu dans le paragraphe précédent, TRIBES peut être considéré comme un algorithme sans paramètres de contrôle. TRIBES est défini comme une boîte noire, pour laquelle l'utilisateur n'a qu'à spécifier le problème à résoudre. Proposer une OEP adaptative impose de répondre à deux types de questions : "Comment la structure de l'essaim évolue au cours du temps ?" et "Quel comportement doit adopter une particule ?". La première question est relative à la définition du paramètre N et de la topologie de voisinage. La deuxième question est relative à la définition des paramètres w, c_1, c_2 et V_{max}. Deux types d'adaptations sont alors définis : les adaptations structurelles et les adaptations comportementales. Ces deux types d'adaptations sont définis dans les sections suivantes.

2.3 Adaptations structurelles

2.3.1 Structure de l'essaim

Dans TRIBES, l'essaim est divisé en plusieurs sous-essaims appelés *tribus*. Les tribus sont de tailles différentes, qui évoluent au cours du temps. Le but est d'explorer simultanément plusieurs régions de l'espace de recherche, généralement des optima locaux, avant de prendre une décision globale. Dans le but de prendre un telle décision, les tribus échangent leurs résultats tout au long du traitement. Un tel type de structure est comparable aux structures multi-essaims utilisées dans d'autres algorithmes [Parsopoulos et al., 2004 ; Niu et al., 2005]. Deux types de communications sont donc à définir : la communication intra-tribu et la communication inter-tribu.

Chaque tribu est composée d'un nombre variable de particules. Les relations entre les particules à l'intérieur d'une tribu sont définies par une topologie complètement connectée. Chaque particule connait la meilleure et la plus mauvaise position jamais atteintes par la tribu, c'est-à-dire que chaque particule connait les positions pour lesquelles la tribu a trouvé la plus petite et la plus grande valeur

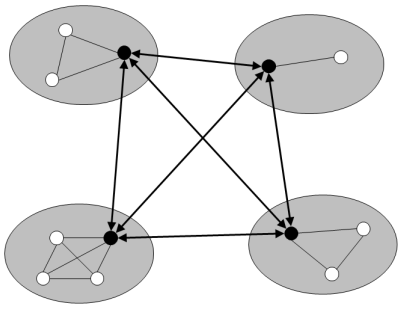


FIGURE 2.1 – Réseau d'information. Chaque tribu est un graphe complètement connecté (communication intra-tribu). Les tribus sont reliées par l'intermédiaire de leurs chamans (communication inter-tribu).

de la fonction objectif. Cette forme de communication est appelée *communication intra-tribu*.

Au fur et à mesure du traitement, chaque tribu va converger vers un optimum local. Il est donc nécessaire que celles-ci communiquent entre elles pour définir lequel de ces optima est à retenir par l'utilisateur. La communication entre les tribus est établie par l'intermédiaire des meilleures particules de chaque tribu. Ce type de communication est appelé *communication inter-tribu*.

En résumé, chaque particule est informée par elle-même (mémoire cognitive $\overrightarrow{p}$), par tous les éléments de sa tribu (appelés *informateurs internes*), et, si cette particule est un *chaman* (e.g. la meilleure particule d'une tribu), alors elle est aussi informée par les chamans des autres tribus (appelés *informateurs externes*). Toutes ces positions sont appelées les *informateurs* de la particule. La mémoire sociale $\overrightarrow{g}$ de chaque particule est l'informateur pour lequel la valeur de la fonction objectif est la plus petite.

La Figure 2.1 illustre la topologie du graphe d'information de l'essaim. Les flèches symbolisent la communication inter-tribu alors que les traits matérialisent la communication intra-tribu. Les particules noires sont les chamans de chaque tribu. On verra dans la suite que cette structure est modifiée automatiquement par l'intermédiaire de créations et de destructions de particules.

2.3.2 Évolution des tribus

Dans le but de définir des règles d'adaptation structurelles pour l'essaim, on définit deux indicateurs de qualité, un pour les particules et un pour les tribus. Une particule est dite *bonne* si elle a amélioré sa meilleure performance au cours de la dernière itération. Dans le cas contraire, elle est dite *neutre*. Cet indicateur est purement qualitatif, car il est basé sur l'évolution de la performance et non sur la performance elle-même. En plus de cet indicateur, la *meilleure* et la *plus mauvaise* position de la tribu sont stockées.

Une tribu est aussi dite *bonne* ou *mauvaise* suivant le nombre de bonnes particules présentes en son sein. Une tribu est dite *mauvaise* si aucune de ses particules n'a amélioré sa meilleure performance au cours de la dernière itération. Les tribus qui possèdent au moins une *bonne* particule sont déclarées *bonnes* avec une probabilité de 0.5, *mauvaises* sinon.

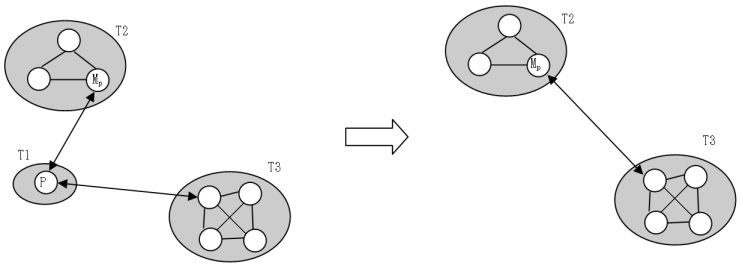


FIGURE 2.2 – Destruction de particule. La tribu *T1* est détruite, les liens d'informations sont redistribués vers *T2* et *T3*.

L'utilisation de ces indicateurs permet de définir les règles de création et de destruction de particules.

2.3.2.1 Destruction de particules

L'élément le plus coûteux en temps lors de l'exécution d'un algorithme d'optimisation est l'évaluation de la fonction objectif. Dans un souci de gain de temps, il est important d'évaluer la fonction objectif le moins de fois possible. Notre intérêt est donc de détruire des particules de l'essaim à chaque fois que cela est possible. Ces destructions sont réalisées en espérant que le résultat final ne soit pas altéré. C'est pourquoi seules les *plus mauvaises* particules des *bonnes* tribus sont détruites. Il est ici considéré que de telles particules n'apportent plus d'informations pertinentes à la tribu, et donc à l'essaim, au regard des performances de ses congénères.

Dans le cas d'une tribu composée d'une seule particule, la destruction est réalisée seulement si l'un des informateurs externes a une meilleure performance que la particule considérée. Cela nous assure de ne retenir que les particules qui apportent de bonnes informations. Sur la Figure 2.2, la tribu *T1* est déclarée *bonne*. La particule *P,* qui est obligatoirement la *plus mauvaise* de la tribu, est détruite, même si elle en est aussi la *meilleure*. Cependant, P ne sera détruite que si son informateur externe M_p possède une meilleure performance. L'hypothèse est que l'information apportée par P est de moins bonne qualité que celle apportée par M_p.

La destruction de particules implique un changement dans le graphe d'information de l'essaim. Toutes les particules informatrices d'une particule supprimée sont redirigées vers la *meilleure* particule de sa tribu. Dans le cas particulier d'une tribu de taille 1, tous les liens d'informations sont redirigés vers le meilleur informateur de la particule supprimée.

2.3.2.2 Création de particules

Le procédé qui permet de créer des particules est assez similaire à celui qui permet d'en détruire. Chaque *mauvaise* tribu génère des particules. Le nombre de particules générées par chaque mauvaise tribu est défini par l'équation (2.1). Cette équation est inspirée d'une formule identique prouvée pour l'OEP binaire [Clerc, 2005]. Toutes les particules générées par les différentes *mauvaises* tribus forment une nouvelle tribu.

$$NB_{particules} = \max\left(2, \left\lfloor \frac{9.5 + 0.124 \cdot (D-1)}{tribeNb} \right\rfloor\right) \tag{2.1}$$

où D est la dimension de l'espace de recherche et $tribeNb$ le nombre de tribus dans l'essaim.

Les *mauvaises* tribus sont alors reliées à la nouvelle tribu et utilisent la nouvelle information apportée par cette tribu pour améliorer leurs performances.

Deux types de particules sont générés, le type de particule étant sélectionné aléatoirement :

- Les particules libres :

Ces particules sont générées aléatoirement à l'aide d'une distribution uniforme soit dans tout l'espace de recherche, soit sur une frontière de celui-ci, ou sur un sommet. La sélection de la méthode de génération se fait aléatoirement parmi les trois possibilités. Les particules sont générées en utilisant (2.2). Le but est d'apporter de la diversité à la population.

$$\begin{cases} \textit{Dans tout l'espace de recherche} : & X_j = U(x_{min-j}, x_{max-j}),\ j \in \{1, \ldots, D\} \\ \textit{Sur une frontière} : & X_j = \begin{cases} U(x_{min-j}, x_{max-j}) & j \in I \subset \{1, \ldots, D\} \\ U\left(\{x_{min-j}, x_{max-j}\}\right) & j \in J \subset \{1, \ldots, D\} \end{cases} \\ \textit{Sur un sommet} : & X_j = U\left(\{x_{min-j}, x_{max-j}\}\right),\ j \in \{1, \ldots, D\} \end{cases} \tag{2.2}$$

où $U(x_{min-j}, x_{max-j})$ est un réel uniformément choisi dans $[x_{min-j}, x_{max-j}]$ et $U\left(\{x_{min-j}, x_{max-j}\}\right)$ est un réel uniformément choisi dans la liste $\{x_{min-j}, x_{max-j}\}$. I et J sont deux sous-espaces qui forment une partition de $\{1, \ldots, D\}$. Ces deux espaces sont définis aléatoirement pour chaque particule générée.

- Les particules confinées :

Si $\overrightarrow{x}$ est la meilleure particule de la tribu génératrice et si $\overrightarrow{i_x}$ est sa meilleure informatrice, $\overrightarrow{p_x}$ et $\overrightarrow{p_{i_x}}$ sont les meilleures performances de $\overrightarrow{x}$ et de $\overrightarrow{i_x}$. La nouvelle particule est générée dans la D-sphère de centre $\overrightarrow{p_{i_x}}$ et de rayon $\|\overrightarrow{p_x} - \overrightarrow{p_{i_x}}\|$ à l'aide de l'équation (2.3). L'objectif est ici d'intensifier la recherche de la tribu génératrice.

$$\overrightarrow{X}_{generated} = alea_{sphere}\left(\overrightarrow{p_{i_x}}, \|\overrightarrow{p_x} - \overrightarrow{p_{i_x}}\|\right) \tag{2.3}$$

où $alea_{sphere}\left(\overrightarrow{p_{i_x}}, \|\overrightarrow{p_x} - \overrightarrow{p_{i_x}}\|\right)$ est un point uniformément choisi dans une hyper-sphère de centre $\overrightarrow{p_{i_x}}$ et de rayon $\|\overrightarrow{p_x} - \overrightarrow{p_{i_x}}\|$.

2.3.3 Fréquence des adaptations

Les adaptations structurelles ne doivent pas être effectuées à chaque itération. En effet, du temps doit être laissé pour que l'information introduite par la dernière modification de la topologie se propage dans l'essaim. Théoriquement, le temps nécessaire pour que l'information se propage dans tout l'essaim est égal au diamètre du graphe d'information. Cela assure que chaque particule soit directement ou indirectement informée des changements apportés lors de la dernière adaptation [Harary, 1994]. Cependant, si la taille de l'essaim devient trop importante, ce nombre peut vite devenir très long à calculer. En pratique, si NL est le nombre de liens d'informations lors de la dernière adaptation, la prochaine adaptation sera effectuée $NL/2$ itérations plus tard. NL est estimé à l'aide de l'équation (2.4).

$$NL = \sum_{n=1}^{tribeNb} explorerNb[n]^2 + tribeNb.(tribeNb - 1) \tag{2.4}$$

Algorithme 9 Adaptations structurelles

test=0
Pour i=1 à *tribeNb* **faire**
  Si *tribe[i].status=bad*
    Générer $NB_{particules} = Max\left(2, \left\lfloor \frac{9.5+0.124.(D-1)}{tribeNb} \right\rfloor\right)$ particules
    test=1
  Fin Si
  Si *tribe[i].status=good*
    Supprimer la *plus mauvaise* particule de tribe[i]
    Si *tribe*[*i*].*size* $\neq 1$
      Rediriger les liens d'information vers la meilleure particule de la tribu
    sinon
      tribeNb=tribeNb-1
      Rediriger les liens d'information vers la meilleure informatrice
    Fin Si
  Fin Si
Fin Pour
Si *test*=1
  tribeNb= tribeNb+1
  Agréger les particules générées à la nouvelle tribu
  Etablir un lien entre les différents chamans
Fin Si
Calculer *NL*

où *tribeNb* est le nombre de tribus et *explorerNb*[*n*] est le nombre de particules de la tribu *n*. Le premier terme du membre de droite de l'équation (2.4) suppose que les particules d'une tribu sont complètement connectées. Donc, pour la tribu *n*, le nombre de liens d'information correspondant à la communication intra-tribu est égal à $explorerNb[n]^2$. Le second terme suppose quant à lui que tous les chamans sont reliés entre eux. Chaque chaman est relié à tous les autres, donc il possède $tribeNb - 1$ liens d'informations correspondant à la communication inter-tribu.

2.3.4 Évolution de l'essaim

Les adaptations structurelles peuvent être résumées par l'Algorithme 9.

Au début du traitement, l'essaim est composé d'une particule unique, qui constitue une tribu à elle seule. Si, durant la première itération, cette particule n'améliore pas sa performance, de nouvelles particules vont alors être générées pour former une deuxième tribu. $NL/2$ itérations plus tard, le même procédé est répété. On continue selon ce schéma durant toute l'exécution.

La taille de l'essaim augmente jusqu'à ce que l'essaim trouve des zones "prometteuses" de l'espace de recherche. Plus la taille de l'essaim devient grande, plus le temps séparant deux adaptations va être long. La capacité d'exploration est améliorée et il va être plus facile pour les particules de trouver de bonnes solutions.

Une fois que des zones intéressantes de l'espace de recherche sont trouvées, les plus mauvaises particules vont être progressivement supprimées.

TABLE 2.1 – Stratégies de déplacement.

Passé	Stratégie de déplacement
(= +) (+ +)	locale par gaussiennes indépendantes
(+ =) (- +)	pivot bruité
(- -) (= -) (+ -) (- =) (= =)	pivot

2.4 Adaptations comportementales

La seconde manière d'adapter le comportement de l'essaim est de proposer différentes stratégies de déplacement des particules, en fonction de leur comportement passé. Le choix se base sur le passé proche de la particule. L'idée est de permettre aux particules qui ont un bon comportement d'intensifier la recherche dans leur zone et, inversement, aux particules qui évoluent mal de pouvoir diversifier leur recherche.

Il y a trois possibilités d'évolution pour décrire l'historique d'une particule entre deux itérations : amélioration, détérioration ou statu quo, suivant que la particule a amélioré, détérioré ou égalé sa performance précédente. Ces trois états sont notés en utilisant les symboles suivants : + pour amélioration, - pour une détérioration et = pour un statu quo. Le passé récent d'une particule est constitué par ses deux dernières variations. Par exemple, une amélioration suivie d'une détérioration est notée (+ -). Il y a donc neuf possibilités de passés possibles. Cependant, on se contente de les diviser en trois groupes (cf. Table 2.1).

(= +) et (+ +) représentent les particules qui ont le meilleur comportement, donc on choisit d'intensifier la recherche dans leur zone à l'aide d'une stratégie locale à gaussiennes indépendantes. (+ =) et (- +) représentent les particules au comportement moyen. Il leur est attribué une stratégie par pivots bruités. Enfin, (- -), (- =), (+ -), (- =) et (= =) représentent les plus mauvaises particules. On leur attribue donc une possibilité de déplacement relativement large, en utilisant une stratégie par pivot.

Les stratégies de déplacement sont définies de la manière suivante :

- Pivot :

Cette méthode est inspirée des travaux publiés dans [Serra et al., 1997]. On note $\overrightarrow{p}$ la meilleure position jamais atteinte par la particule, $\overrightarrow{g}$ sa meilleure informatrice et f la fonction objectif. Le mouvement est déterminé par :

$$\overrightarrow{X} = c_1 \cdot alea_{sphere}\,(H_p) + c_2 \cdot alea_{sphere}\,(H_g) \tag{2.5}$$

où $c_1 = \frac{f(\overrightarrow{p})}{f(\overrightarrow{p})+f(\overrightarrow{g})}$, $c_2 = \frac{f(\overrightarrow{g})}{f(\overrightarrow{p})+f(\overrightarrow{g})}$, $alea_{sphere}(H_p)$ est un point uniformément choisi dans l'hyper-sphère de centre $\overrightarrow{p}$ et de rayon $\|\overrightarrow{p} - \overrightarrow{g}\|$ et $alea_{sphere}(H_g)$ est un point uniformément choisi dans l'hyper-sphère de centre $\overrightarrow{g}$ et de rayon $\|\overrightarrow{p} - \overrightarrow{g}\|$.

- Pivot bruité :

Cette stratégie est identique à la précédente, mais un bruit est ajouté.

$$\overrightarrow{X} = c_1 \cdot alea_{sphere}\,(H_p) + c_2 \cdot alea_{sphere}\,(H_g) \tag{2.6}$$

Algorithme 10 Algorithme de TRIBES

Initialisation aléatoire de la particule initiale $\overrightarrow{X_0}$
$\overrightarrow{p_0} = \overrightarrow{g_0} = \overrightarrow{X_0}$
Evaluer la fonction objectif
Tant que le critère d'arrêt n'est pas atteint, **faire**
  Déterminer le statut de chaque particule
  Choisir la stratégie de déplacement
  Déplacement des particules
  Evaluer la fonction objectif
  Mise à jour des $\overrightarrow{p_i}$ et des $\overrightarrow{g_i}$
  Si $n<NL$
    Déterminer la qualité des tribus
    Adaptations structurelles
    Calculer NL
  Fin Si
Fin Tant que

$$b = N\left(0,\ \frac{f(\overrightarrow{p}) - f(\overrightarrow{g})}{f(\overrightarrow{p}) + f(\overrightarrow{g})}\right) \tag{2.7}$$

$$\overrightarrow{X} = (1+b)\cdot\overrightarrow{X} \tag{2.8}$$

- Locale par gaussiennes indépendantes :

Si $\overrightarrow{g}$ est la meilleure informatrice de la particule, le mouvement est déterminé comme suit :

$$X_j = g_j + alea_{normal}\left(g_j - X_j, \|g_j - X_j\|\right),\ j \in \{1,\ldots,D\} \tag{2.9}$$

où $alea_{normal}(g_j\text{-}X_j,\ \|g_j - X_j\|)$ est un point aléatoirement choisi à l'aide d'une distribution gaussienne de moyenne g_j-X_j et de variance $\|g_j - X_j\|$.

Il est important de noter que, contrairement à l'OEP originale, le concept de vitesse n'apparaît pas dans TRIBES. Il est précisé dans [Kennedy, 2003] que la suppression de ce concept n'est pas un obstacle à l'OEP. La seule différence notable est que la distribution de probabilité de la prochaine position est modifiée, elle est D-sphérique dans le cas de TRIBES et D-parallélépipédique dans le cas de l'OEP classique [Clerc, 2006]. Ceci implique que, dans TRIBES, le déplacement n'est pas dépendant du système de coordonnées. La philosophie globale de l'algorithme est toutefois respectée, le vecteur vitesse pouvant être modélisé par $\overrightarrow{X}(t+1) - \overrightarrow{X}(t)$.

2.5 Algorithme

L'algorithme de TRIBES est donné par l'Algorithme 10. $\overrightarrow{g_i}$ est la meilleure informatrice de la particule i, $\overrightarrow{p_i}$ est la meilleure position atteinte par la particule i. n est le nombre d'itérations effectuées depuis la dernière adaptation structurelle.

2.6 Expérimentation numérique et comparaisons

Dans cette section, nous présentons les résultats numériques obtenus par TRIBES, ainsi que des comparaisons avec d'autres algorithmes. L'intérêt est de montrer que TRIBES donne des résultats encourageants. Une discussion sur les avantages et inconvénients de TRIBES est aussi réalisée.

2.6.1 Procédure de test CEC'05

La procédure de test utilisée pour comparer les algorithmes est celle qui a été définie dans le cadre de la conférence *2005 IEEE Congress on Evolutionary Computation* (CEC'05) [Suganthan et al., 2005]. Le but de cette procédure est d'uniformiser les tests effectués sur les métaheuristiques et, ainsi, de faciliter les comparaisons entre celles-ci. En effet, dans les différents articles sur le sujet, les algorithmes ne sont jamais évalués de la même façon, donc il est souvent difficile de comparer deux algorithmes sans avoir à les coder. Posséder une procédure de test commune permet donc de faciliter les comparaisons entre les différentes méthodes.

Lors de la conférence CEC'05 a été défini un ensemble de 25 fonctions de test. Ces fonctions ont été choisies ou créées afin d'éviter certaines topologies de l'espace de recherche [Liang et al., 2005b]. En effet, il a été montré que certaines particularités topologiques des fonctions objectifs peuvent favoriser certains algorithmes. Par exemple, il existe des algorithmes qui ont la capacité de converger plus rapidement si l'optimum global est situé à l'origine $[0, 0, \ldots, 0]$ de l'espace de recherche ou au centre de celui-ci [Zhong et al., 2004]. Pour éviter cela, les fonctions définies lors de CEC'05 sont translatées par rapport à leur forme d'origine et/ou tournées. Certaines fonctions présentent aussi la particularité d'avoir leur optimum global situé sur une des frontières de l'espace de recherche. Deux types de tests sont définis : un qui étudie la valeur de l'erreur après un nombre fixé d'évaluations de la fonction objectif et un qui étudie le nombre d'évaluations de la fonction objectif nécessaires pour obtenir une précision donnée.

Le premier test consiste à exécuter le même algorithme 25 fois sur chacun des problèmes. Un critère d'arrêt unique est utilisé : l'algorithme est stoppé si le nombre d'évaluations de la fonction objectif dépasse $Max_{Fes} = 10000 \cdot D$, où D est la dimension du problème. La valeur de l'erreur entre la meilleure particule de l'essaim et l'objectif est enregistrée pour 10^3, 10^4, 10^5 et Max_{Fes} évaluations de la fonction objectif. Les résultats sur les 25 exécutions sont alors classés, la moyenne et la variance de l'erreur sont calculées. Ces résultats sont utilisés pour construire le graphe de convergence de l'algorithme. Le graphe de convergence trace la valeur médiane de l'erreur sur les 25 exécutions en fonction du nombre d'évaluations de la fonction objectif. Une échelle semi-logarithmique est utilisée.

Le deuxième test consiste à déterminer le nombre d'évaluations nécessaires pour obtenir une précision donnée. L'algorithme est stoppé si l'erreur entre la meilleure particule de l'essaim et l'objectif est inférieure à une précision donnée ou si le nombre d'évaluations de la fonction objectif est supérieur à Max_{Fes}. Les résultats sont enregistrés pour chaque exécution et classés. On définit aussi deux indicateurs de performances :

$$Taux\,de\,succes = \frac{nombre\,d'executions\,reussies}{nombre\,total\,d'executions} \tag{2.10}$$

$$Taux\,de\,Performance = \frac{Fes_{mean} \times nombre\,total\,d'executions}{nombre\,d'executions\,reussies} \tag{2.11}$$

où Fes_{mean} est le nombre moyen d'évaluations de la fonction objectif pour les exécutions réussies. Les précisions utilisées sont indiquées dans la Table 2.2.

TABLE 2.2 – Précisions utilisées comme critère d'arrêt pour le test n°2.

Fonction	1	2	3	4	5	6	7	8	9	10	11
Précision	1e-6	1e-6	1e-6	1e-6	1e-6	1e-2	1e-2	1e-2	1e-2	1e-2	1e-2

2.6.2 Algorithmes utilisés pour la comparaison

TRIBES est comparé aux algorithmes de la Table 2.3. Dans ce paragraphe, une description succincte de chacun de ces algorithmes est donnée. BLX-GL50 est un algorithme génétique à variables continues hybridé avec une stratégie de différenciation mâle/femelle [Garcia-Martinez et al., 2005]. BLX-MA [Molina et al., 2005] est un algorithme mimétique à variables continues, qui combine une exploration globale de l'espace de recherche avec une recherche locale adaptative qui favorise les meilleurs éléments. CoEvo [Posik, 2005] est un algorithme évolutionnaire à variables réelles. Il utilise une stratégie coopérative d'évolution qui permet de reproduire les mutations qui ont eu un bon effet sur les performances. DE est un algorithme d'évolution différentielle simple [Rönkkonën et al., 2005]. DMS-L-PSO est un algorithme d'OEP multi-essaim. L'essaim est divisé en sous-essaims, qui échangent de l'information mais qui peuvent aussi être regroupés [Liang et al. 2005c]. Une méthode de type Newton est utilisée comme recherche locale. EDA est un algorithme à estimation de distribution basique [Yuan et al., 2005]. G-CMA-ES est un algorithme évolutionnaire avec une stratégie de "redémarrage" adaptable à l'aide d'une matrice de covariance [Auger et al., 2005]. K-PCX est un algorithme à population développé par Sinha et al. [Sinah et al., 2005]. L-CMA-ES est une hybridation de CMA-ES avec un algorithme de recherche locale [Auger et al., 2005b]. L-SaDE est un algorithme d'évolution différentielle adaptatif disposant de deux méthodes d'apprentissage [Qin et al., 2005]. SPC-PNX est un algorithme génétique à variables continues disposant d'opérateurs de sélection, de croisement et de remplacement particuliers [Ballester et al., 2005]. Les articles relatifs à ces méthodes sont téléchargeables gratuitement à l'adresse suivante [Papers, 2005]. Enfin, on compare TRIBES avec l'algorithme *Standard PSO 2006*, qui est un algorithme d'OEP avec facteur de constriction qui utilise une topologie de voisinage aléatoire. La taille de l'essaim est définie par $N = \left\lfloor 10 + 2\sqrt{D} \right\rfloor$ [PSC, 2006]. Les équations de déplacement sont identiques à (1.5) et (1.6), avec $w = \frac{1}{2.\ln(2)}$ et $c_1 = c_2 = \frac{1}{2} + \ln(2)$. Tous les algorithmes sont listés dans la Table 2.3.

2.6.3 Résultats numériques

Toutes les tables citées dans cette section sont situées dans l'annexe B.

Les résultats numériques présentés dans cette section ont été obtenus grâce à la version de TRIBES codée dans notre laboratoire. Les résultats obtenus pour les algorithmes présentés en 2.6.2 ont été tirés des articles référencés dans la Table 2.3, à l'exception du *Standard PSO 2006*, pour lequel le code source donné à l'adresse [PSC,2006] est utilisé. La Table B.1 présente la valeur absolue de l'erreur en dimension 10 pour les fonctions unimodales, la Table B.2 présente la valeur absolue de l'erreur en dimension 10 pour les fonctions multimodales simples et, la Table B.3 présente la valeur absolue de l'erreur en dimension 10 pour les fonctions étendues et composées.

Les fonctions F15 à F25 sont des fonctions composées, i.e. leurs expressions mathématiques sont des sommes pondérées, biaisées et tournées des fonctions F1 à F14 [Suganthan et al., 2005].

Les Tables B.4, B.5 et B.6 offrent une comparaison statistique entre les résultats de TRIBES donnés dans les Tables B.1 et B.2, et les résultats des algorithmes de la Table 2.3. Ces statistiques sont établies pour les fonctions F1 à F11. L'hypothèse du test est : "Les résultats donnés par TRIBES sont meilleurs que ceux donnés par l'algorithme X". Nous supposons ici que, pour un nombre d'éva-

TABLE 2.3 – Algorithmes utilisés pour les comparaisons.

Algorithme	**Nom**	**Référence**
Algorithme génétique à variables continues hybride avec différenciation Mâle/Femelle	BLX-GL50	[García-Martínez et al., 2005]
Algorithme mimétique à variables continues	BLX-MA	[Molina et al., 2005]
Optimisation à variables continues utilisant une mutation co-évolutive	CoEVO	[Posik, 2005]
Optimisation à variables continues avec évolution différentielle	DE	[Rönkkönen et al., 2005]
Optimisation par essaim particulaire multi-essaim dynamique avec recherche locale	DMS-L-PSO	[Liang et al., 2005c]
Algorithme à estimation de distribution à variables continues	EDA	[Yuan et al., 2005]
Algorithme CMA avec stratégie de redémarrage et taille de population variable	G-CMA-ES	[Auger et al., 2005]
Procédure *steady-state* pour l'optimisation à variables continues	K-PCX	[Sinha et al., 2005]
Algorithme CMA avec recherche locale	L-CMA-ES	[Auger et al., 2005b]
Algorithme d'évolution différentielle adaptative	L-SaDE	[Qin et al., 2005]
Algorithme génétique à variables continues avec procédure *steady-state*	SPC-PNX	[Ballester et al., 2005]
Standard PSO 2006	SPSO	[PSC, 2006]

luations de la fonction objectif donné, les résultats de TRIBES suivent une distribution gaussienne tronquée (bornée à gauche par zéro) $N_1(\mu_1, \sigma_1)$ et que ceux de l'algorithme X suivent une distribution gaussienne tronquée (bornée à gauche par zéro) $N_2(\mu_2, \sigma_2)$. Les résultats expérimentaux nous donnent m_1, s_1, m_2 et s_2, qui sont des approximations de μ_1, σ_1, μ_2 et σ_2. La probabilité que l'hypothèse soit vraie, i.e. la probabilité que $\mu_1 < \mu_2$, est donnée par:

$$p(m_1 < m_2) = \int_0^{+\infty} x.N_1(x).\left(1 - N_{2c}(x)\right).dx \tag{2.12}$$

où $N_{2c}(u)$ est la probabilité qu'un nombre tiré aléatoirement en utilisant N_2 soit plus petit que u.

La Table B.7 présente les résultats de TRIBES pour le test n°2.

La Table B.8 présente une comparaison entre TRIBES et les algorithmes de la Table 2.3 sur les fonctions unimodales en dimension 10 pour le test n°2. La première ligne donne le meilleur *Taux de Performance* trouvé et le reste du tableau donne les *Taux de Performance* normalisés. Les chiffres entre parenthèses donnent les nombres d'exécutions ayant abouti à un succès.

La Table B.9 est structurée comme la Table 2.11, mais les résultats pour les fonctions multimodales sont cette fois-ci présentés.

Les Tables B.10 et B.11 donnent les résultats de TRIBES en dimension 30, pour les fonctions unimodales et multimodales, respectivement.

Les Tables B.12 et B.13 donnent les résultats de TRIBES en dimension 50 pour les fonctions unimodales et multimodales, respectivement.

Les Tables B.14 et B.15 comparent les résultats moyens obtenus par TRIBES et les algorithmes de la Table 2.3 en dimensions 30 et 50.

2.6.4 Discussion

Les Tables B.1 à B.15 et les Figures 2.3 à 2.11 illustrent le comportement de TRIBES.

2.6.4.1 Commentaires généraux

A la vue de ces résultats expérimentaux, il apparaît que TRIBES présente un défaut majeur : la convergence prématurée. Les Figures 2.3, 2.4 et 2.5 montrent les graphes de convergence pour les fonctions Rosenbrock (F6), Griewank (F7) et Rastrigin (F9). Ces graphes représentent l'erreur médiane sur les 25 exécutions, en fonction du nombre d'évaluations de la fonction objectif (FEs). Ces trois cas, qui sont représentatifs du comportement général, montrent que l'erreur décroît assez rapidement jusqu'à approximativement 10000 évaluations de la fonction objectif mais, qu'ensuite, elle a tendance à ne plus trop évoluer.

Les Figures 2.6, 2.7 et 2.8 montrent l'évolution de la position de la meilleure particule de l'essaim en fonction du nombre d'itérations pour les fonctions Rosenbrock (F6), Griewank (F7) et Rastrigin (F9). Comme sur les figures précédentes, on voit que la meilleure particule de l'essaim converge rapidement vers une position de l'espace de recherche et ne bouge pas beaucoup après.

On est donc ici dans le cas où la meilleure particule de l'essaim est "piégée" dans un optimum local et n'arrive pas à s'en échapper. Il en découle que les autres particules de l'essaim sont aussi attirées vers cet optimum local et, vont donc se faire piéger les unes après les autres dans cet optimum local. A partir de ce moment là, la phase de diversification devient inexistante. L'algorithme est alors en situation d'échec. Cependant, sur la Figure 2.8, on peut voir, aux alentours de l'itération 1000, que l'ajout de nouvelles particules permet une évolution significative de la meilleure position de l'essaim. La Figure 2.9 montre la trajectoire d'une particule dans le cas de la fonction Weierstrass 2D (F11). Il apparaît bien ici comment la particule "stagne" à l'intérieur de la vallée d'un optimum local. Les

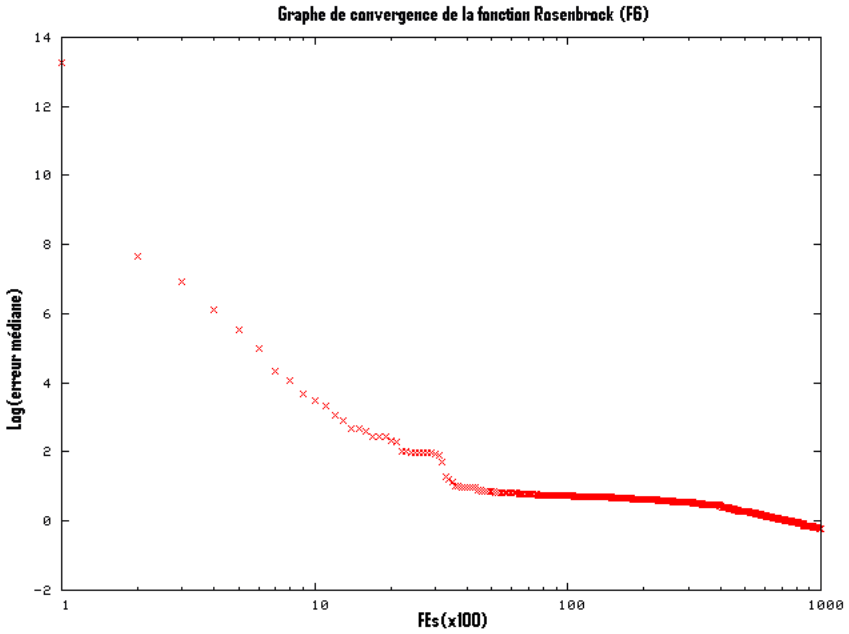


FIGURE 2.3 – Graphe de convergence de la fonction Rosenbrock (F6).

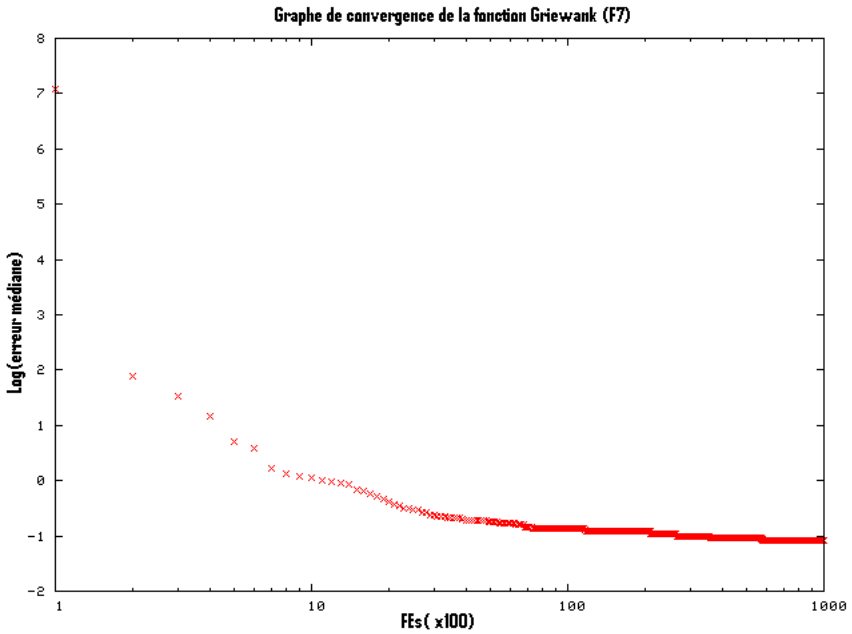


FIGURE 2.4 – Graphe de convergence de la fonction Griewank (F7).

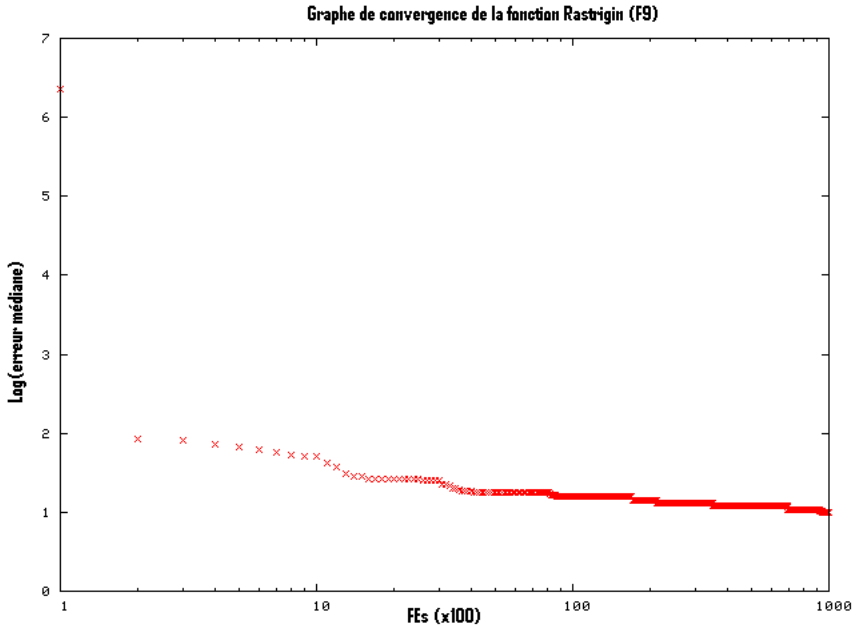


FIGURE 2.5 – Graphe de convergence de la fonction Rastrigin (F9).

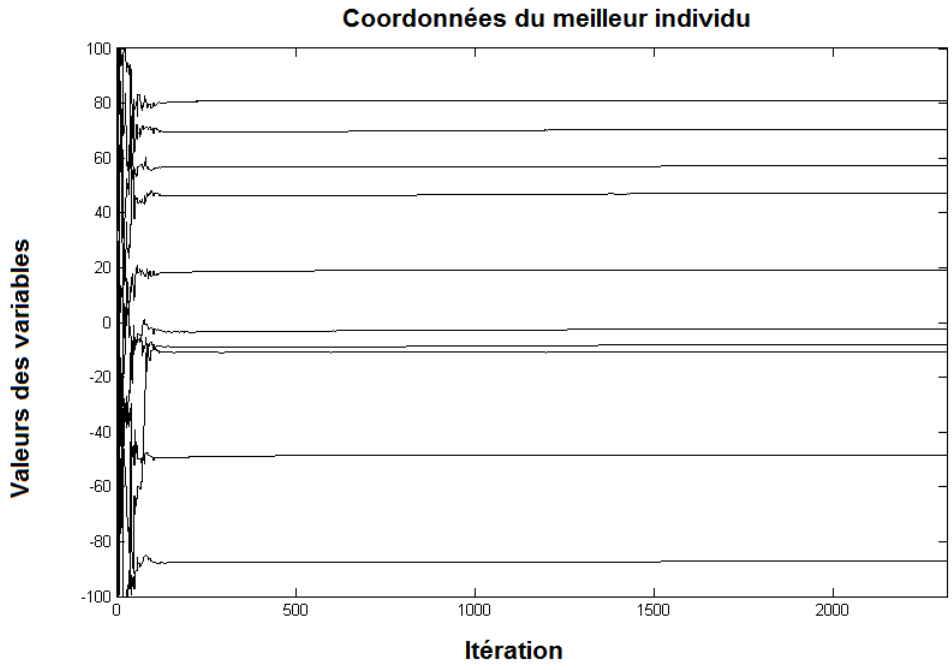


FIGURE 2.6 – Vecteur position de la meilleure particule de l'essaim pour la fonction Rosenbrock (F6) en dimension 10.

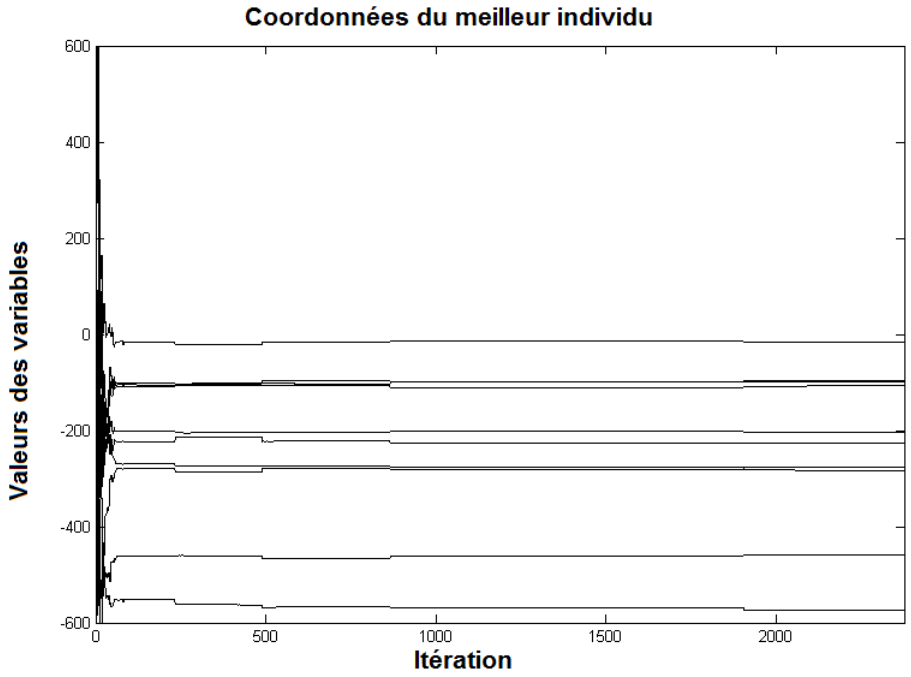


FIGURE 2.7 – Vecteur position de la meilleure particule de l'essaim pour la fonction Griewank (F7) en dimension 10.

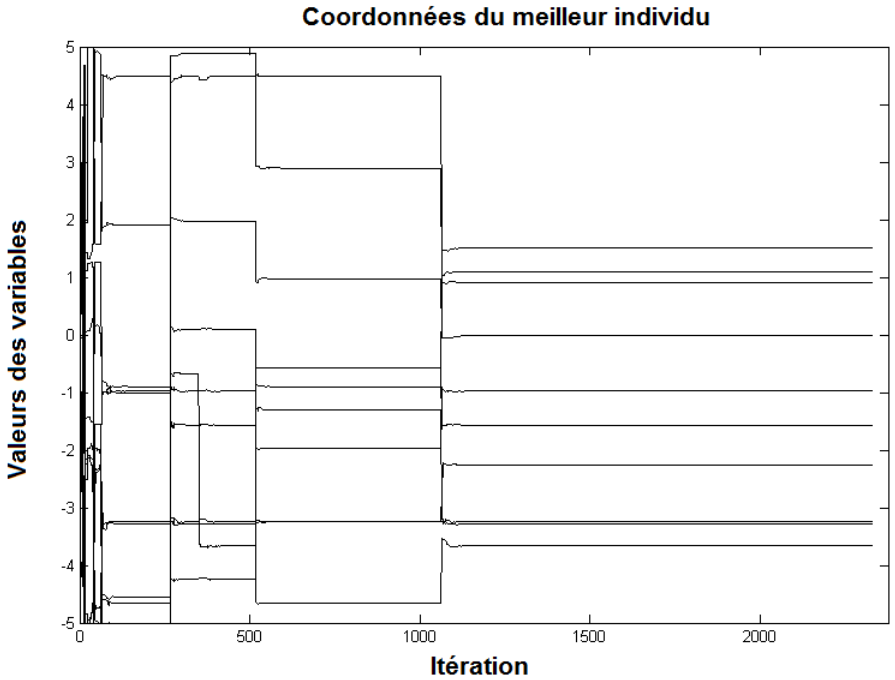


FIGURE 2.8 – Vecteur position de la meilleure particule de l'essaim pour la fonction Rastrigin (F9) en dimension 10.

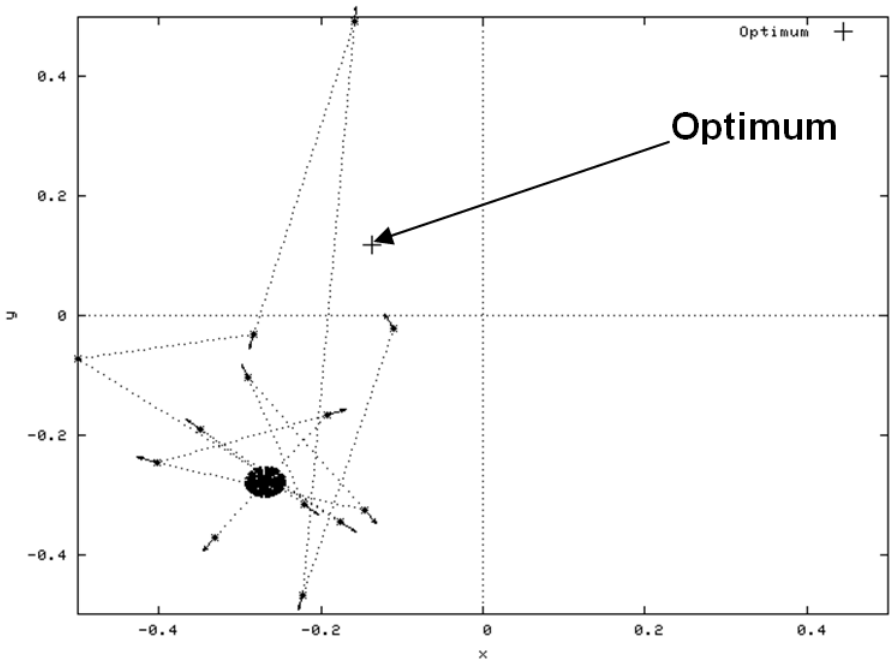


FIGURE 2.9 – Trajectoire d'une particule pour la fonction Weierstrass 2D (F11).

flèches représentent les vecteurs $\overrightarrow{X_i}(t) - \overrightarrow{X_i}(t-1)$, $i \in \{1, \ldots, N\}$, qui modélisent le vecteur vitesse de la particule à chaque instant.

L'observation des vecteurs vitesse à la fin de l'exécution nous montre que la particule a tendance à emprunter une trajectoire "circulaire" autour de l'optimum local dans lequel elle est piégée. Ce type d'inconvénient est inhérent au paradigme de l'OEP. Le déplacement d'une particule est influencé par sa mémoire cognitive $\overrightarrow{p}$ et par sa mémoire sociale $\overrightarrow{g}$. Les particules sont donc attirées vers les positions $\overrightarrow{p}$ et $\overrightarrow{g}$. Cela implique que, si $\overrightarrow{p}$ ou $\overrightarrow{g}$ est localisé dans la vallée d'un optimum local, les particules vont être attirées dans celle-ci. Les particules sont alors piégées, tournent autour de cette vallée et n'arrivent plus à améliorer leur performance. Dans TRIBES, l'ajout de particule est censé contrer cet inconvénient. La Figure 2.10 montre l'évolution de la performance de chaque particule en fonction du nombre d'itérations, pour le problème Rastrigin 10D (F9).

Sur cette figure, on peut observer la fréquence des adaptations structurelles au cours du temps. A la fin de l'exécution, l'essaim est composé de 64 particules réparties en 8 tribus. La convergence rapide des particules est ici facilement observable. Il apparaît aussi que, dans ce cas là, l'ajout de nouvelles particules n'apporte aucune aide à l'essaim pour trouver des régions prometteuses de l'espace de recherche. Les meilleures performances de chaque tribu sont quasi-similaires, situées ici dans l'intervalle [2.26, 2.56]. La Figure 11 montre l'histogramme des distances entre toutes les positions atteintes par les particules de l'essaim pour la fonction Rastrigin 10D (F9).

L'histogramme est composé de quatre pics assez étroits. Cela signifie que, à la fin de l'exécution, l'essaim a exploré quatre régions différentes de l'espace de recherche. Le fait que les pics soient très étroits implique que les particules sont concentrées autour d'optima locaux et qu'elles n'arrivent pas à s'échapper de leurs vallées.

Les données numériques des Tables B.1, B.2 et B.3 confirment ce qui a été dit précédemment. TRIBES est performant pour les problèmes Sphere (F1) ou Schwefel (F2) (même dans le cas bruité *Schwefel noise* (F4), où les optima locaux sont peu "profonds"). Cependant, pour les problèmes plus

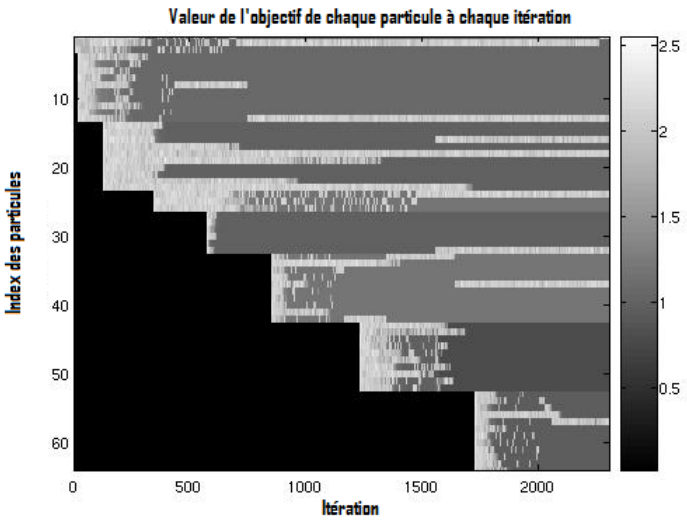


FIGURE 2.10 – Performances des particules en fonction du nombre d'itérations pour la fonction Rastrigin 10D (F9).

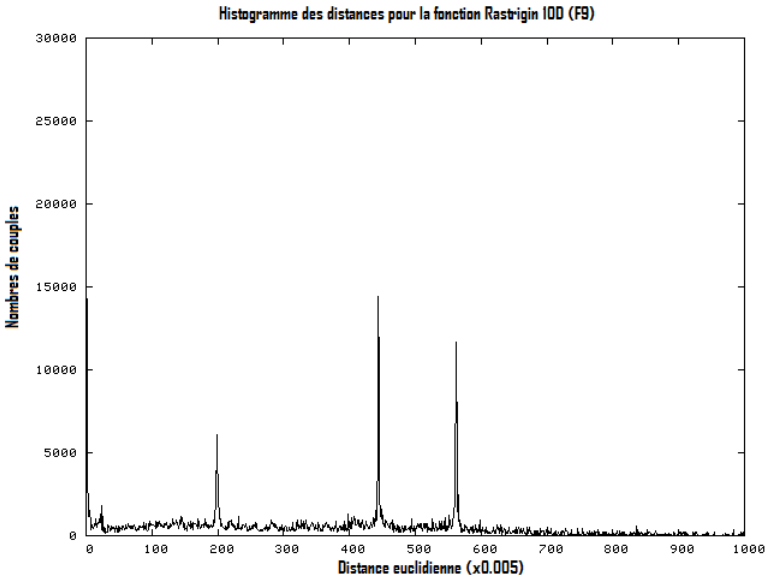


FIGURE 2.11 – Histogramme des distances entre particules pour la fonction Rastrigin 10D (F9).

complexes, TRIBES donne de moins bons résultats. On constate que, entre FEs=10^4 et FEs=10^5, les performances de l'algorithme n'évoluent que très peu, ce qui confirme l'hypothèse d'une stagnation des particules dans les vallées d'optima locaux.

2.6.4.2 Comparaison avec le *Standard PSO 2006*

TRIBES n'aurait aucun intérêt si ses performances étaient inférieures à celles données par un algorithme d'OEP classique. Les Tables B.4 à B.6 montrent que le *Standard PSO 2006* a une probabilité supérieure à 50% d'être meilleure que TRIBES dans seulement 9 cas sur 33. Les Tables B.8 et B.9 montrent que, pour atteindre une précision donnée, TRIBES présente des *Taux de succès* supérieurs à ceux du *Standard PSO 2006*. Cependant, pour les problèmes aboutissant à des succès, les *Taux de Performance* du *Standard PSO 2006* sont souvent meilleurs que ceux de TRIBES. Ceci peut être expliqué par le fait que, en dimension 10, le *Standard PSO 2006* commence le traitement avec un essaim de taille 16 alors que TRIBES ne commence qu'avec une seule particule. Il est donc beaucoup plus simple au *Standard PSO 2006* d'atteindre une précision donnée rapidement qu'à TRIBES. En conclusion, on peut dire que TRIBES présente des performances au moins équivalentes et souvent meilleures que celles du *Standard PSO 2006*, même si la convergence est souvent moins rapide.

2.6.4.3 Comparaison entre algorithmes

Dans l'objectif d'établir des comparaisons entre TRIBES et les algorithmes de la Table 2.3, on exclut la fonction *Ackley bounds* (F8), parce que tous les algorithmes échouent de la même manière pour ce problème et présentent des résultats similaires.

Les Tables B.4 à B.6 confirment nos précédentes remarques sur la convergence prématurée. Pour FEs=10^3, TRIBES présente des résultats qui sont bien meilleurs que ceux des autres algorithmes avec une forte probabilité. La plupart d'entre elles sont supérieures à 90%. Il en est déduit que TRIBES peut trouver très rapidement des zones "prometteuses" de l'espace de recherche. Pour FEs=10^4, aucune tendance claire ne se dégage, mais on voit que TRIBES n'est plus aussi dominateur vis-à-vis des autres algorithmes. En revanche, pour FEs=10^5, les résultats sont très défavorables à TRIBES. On en déduit que des régions "prometteuses" sont effectivement trouvées, mais que l'algorithme n'a pas la capacité de s'extraire des vallées des optima locaux trop profondes.

Les Tables B.7 à B.9 nous informent sur le nombre d'évaluations de la fonction objectif nécessaires à TRIBES pour atteindre une précision donnée. Les problèmes unimodaux sont très bien résolus, avec un *Taux de succès* de 98%. L'observation des *Taux de performance* nous montre que TRIBES n'est, certes, jamais l'algorithme le plus rapide, mais qu'il occupe toujours une bonne place dans celle-ci (2^{nd}, $5^{\text{ème}}$, $4^{\text{ème}}$, $4^{\text{ème}}$ et $8^{\text{ème}}$, respectivement). Même si le traitement ne commence qu'avec une seule particule, TRIBES possède une convergence assez rapide dans le cas des fonctions unimodales. En revanche, l'algorithme est déficient pour les problèmes multimodaux. Le *Taux de succès* est de 1%, ce qui fait de TRIBES l'un des plus mauvais algorithmes. Seul CoEvo présente de plus mauvais résultats. Seules les fonctions F6 et F7 sont résolues et encore seulement une fois sur les 25 exécutions. Les fonctions F6 à F11 possédant de nombreux optima locaux, souvent profonds, il est difficile pour TRIBES d'obtenir de bons résultats sur ces problèmes, eu égard à ce qui a été dit précédemment.

2.6.4.4 Influence de la dimension

Le but de ce paragraphe est de mesurer l'influence de la dimension du problème sur les performances de TRIBES. La comparaison est établie à partir des performances de TRIBES sur les

fonctions F1 à F12 en dimensions 10, 30 et 50 (Tables B.1, B.2, B.10, B.11, B.12, B.13). TRIBES est exécuté en utilisant $Max_{Fes} = 10000 \cdot D$ comme critère d'arrêt.

En dimension 10, TRIBES donne de très bons résultats pour les problèmes unimodaux (F1 à F5). Plus la dimension augmente, plus les résultats se dégradent, même si Max_{Fes} augmente. Les performances ne sont constantes que pour le problème Sphere (F1), qui est le plus simple de tous. De très bons résultats sont aussi trouvés pour le problème Schwefel (F2). Pour les problèmes F3 à F5, les performances de TRIBES se dégradent fortement avec l'augmentation de la dimension. Ces problèmes sont résolus avec une précision aux alentours de 10^{-10} en dimension 10, mais la précision chute à 10^3 pour les dimensions supérieures. Pour le problème F3, la difficulté vient du fait que la fonction a un éventail de valeurs possibles très large, qui augmente fortement avec la dimension. Pour la fonction F4, la difficulté provient du bruit additif. En effet, accroître la dimension implique une augmentation du rapport signal à bruit, donc une augmentation de la complexité du problème. Pour le problème F5, l'optimum global est situé sur une frontière de l'espace de recherche. Il est donc beaucoup plus difficile, pour l'algorithme, de le localiser, si la dimension de l'espace de recherche augmente.

Dans le cas des problèmes multimodaux, les conclusions sont similaires. Dans la majorité des cas, les performances se dégradent avec l'augmentation de la dimension, mais dans des proportions moindres que dans le cas unimodal. La fonction F8 est singulière dans le sens où, quelle que soit la dimension, TRIBES présente toujours les mêmes résultats. Ceci est dû à la topologie de la fonction. Quelle que soit la dimension, les valeurs de la fonction présentent des fluctuations autour de -120 dans tout l'espace de recherche et un pic très fin qui tombe à -140 [Suganthan et al., 2005]. L'erreur, quasi-constante avec l'augmentation de la dimension, provient donc du fait qu'aucune particule ne tombe jamais dans ce pic. Le problème F7 est aussi singulier, dans le sens où TRIBES est de plus en plus efficace à mesure que la dimension augmente. On peut penser que des performances identiques pourraient être obtenues en dimension 10 avec une légère augmentation de Max_{Fes}.

Les Tables B.14 et B.15 montrent une comparaison des erreurs moyennes pour les fonctions F1 à F12. Vis-à-vis des autres algorithmes, le comportement de TRIBES ne change pas avec l'augmentation de la dimension. TRIBES possède généralement des performances qui le situent en milieu de hiérarchie. Le taux de détérioration de la performance avec l'augmentation de la dimension est similaire pour TRIBES et pour les autres algorithmes. Ceci implique que la détérioration des performances de TRIBES est plus due à l'augmentation de la difficulté des problèmes qu'à un dysfonctionnement de TRIBES en haute dimension. Excepté pour le problème F6, TRIBES reste comparable au *Standard PSO 2006* en matière de performances.

2.6.5 Conclusion

TRIBES est un algorithme novateur dans le domaine des métaheuristiques adaptatives, car c'est un algorithme qui n'a aucun paramètre de contrôle à régler. Aucune intervention humaine n'est sollicitée ; seul le problème à résoudre est à spécifier. Ceci est permis par l'utilisation de règles d'adaptation suffisamment robustes. Comparé à un algorithme d'OEP classique, TRIBES donne des résultats compétitifs, que ce soit sur des problèmes unimodaux ou multimodaux, et quelle que soit la dimension. En revanche, comparé à d'autres algorithmes, TRIBES donne des résultats assez faibles sur les problèmes multimodaux. L'algorithme trouve très rapidement des zones "intéressantes" de l'espace de recherche. Dans le cas où ces zones seraient en fait des optima locaux, le problème est que TRIBES n'arrive pas à extraire les particules de l'essaim de ceux-ci.

Converger rapidement vers des zones "intéressantes" de l'espace de recherche est, en soi, un inconvénient, mais il peut se transformer en avantage, dans le cas de l'optimisation dynamique, ou pour les fonctions objectif très longues à évaluer (souvent issues d'un processus expérimental).

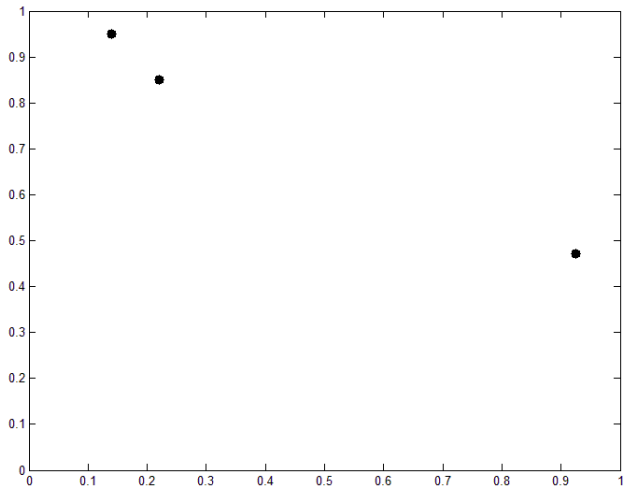


FIGURE 2.12 – Initialisation aléatoire de particules dans un espace de recherche 2D.

Ce travail a fait l'objet de la soumission d'un article à la revue *Swarm Intelligence* [Cooren et al., 2008].

2.7 Amélioration de TRIBES : initialisation et déplacement des particules

2.7.1 Initialisation régulière

Les performances des méthodes d'OEP sont directement liées à l'initialisation des particules. Dans [Campana et al., 2006], il est démontré que, dans le cas d'un algorithme d'OEP classique, la position d'une particule au temps t peut être décomposée en deux vecteurs, l'un qui ne dépend que de la position initiale de la particule et l'autre qui est indépendant de la position initiale de la particule. La trajectoire d'une particule est donc dépendante de sa position initiale et de sa vitesse initiale. De ce fait, il apparaît comme primordial de bien initialiser les particules de l'essaim, afin d'obtenir de meilleurs résultats. L'idéal serait d'initialiser les particules afin que l'espace de recherche soit exploré de la meilleure des manières, i.e. que toutes les régions de celui-ci soient visitées au moins une fois par une des particules de l'essaim.

Dans la grande majorité des cas, les particules sont initialisées aléatoirement dans l'espace de recherche. Cependant, une telle initialisation peut entraîner un manque de diversité dans l'essaim dès le premier pas de temps, ce qui favorise une convergence prématurée vers un optimum local. La Figure 2.12 montre l'exemple d'une initialisation aléatoire de trois particules dans un espace de recherche 2D. On peut observer que les particules sont "concentrées" dans la partie supérieure gauche et sur la frontière de droite de l'espace de recherche. De ce fait, ce sont ces deux régions qui vont d'abord être explorées par les particules. Il y a donc une possibilité que les particules soient piégées dans l'un des optima locaux présents dans l'une de ces deux régions, sans qu'elles aient pu auparavant explorer le reste de l'espace de recherche.

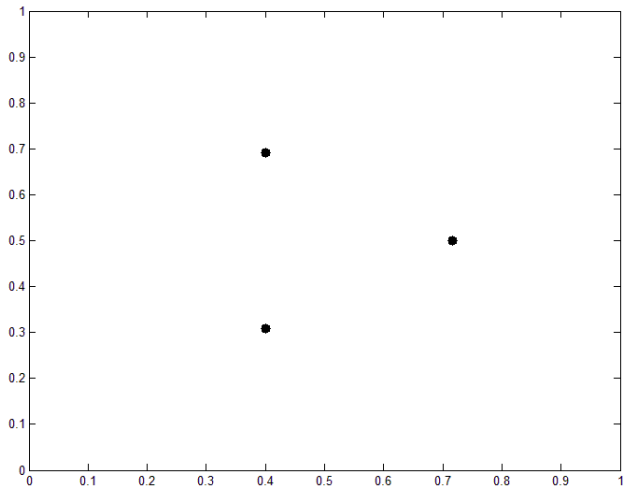


FIGURE 2.13 – Initialisation régulière de particules dans un espace de recherche 2D.

Dans l'objectif de résoudre ce problème, on introduit un nouveau mode d'initialisation. Le but est de couvrir l'espace de recherche le plus uniformément possible avec les particules, dès leur initialisation. En pratique, les particules sont initialisées de manière à être le plus éloigné possible les unes des autres et le plus éloignées possible des frontières de l'espace de recherche. Les particules sont choisies loin des frontières car cela de permet d'éviter les restrictions dans les directions de recherche. Le critère qui permet de choisir la meilleure configuration, en accord avec ce qui a été dit précédemment, est défini comme suit :

$$f = \sum_{i=1}^{nbParticules} \sum_{j \neq i} \frac{1}{d_{ij}} + \sum_{i=1}^{nbParticules} \frac{1}{\min_{d \in \{1 \ldots D\}} \left(d\left(x_i, bound_d\right)\right)} \tag{2.13}$$

où d_{ij} est la distance entre la particule i et la particule j, et $d\left(x_i, bound_d\right)$ est la distance entre la particule i et la frontière de la dimension d.

La Figure 2.13 montre l'exemple d'une initialisation régulière réalisée avec sept particules dans un espace de recherche à deux dimensions.

Dans la nouvelle version de TRIBES, l'initialisation des particules est donc réalisée en initialisant $D + 1$ particules en satisfaisant le critère de l'équation (2.13).

2.7.2 Stratégie de déplacement

L'étude du comportement des particules est un sujet difficile à aborder. En effet, les différentes interactions entre les particules et le caractère aléatoire des déplacements rendent l'étude théorique des trajectoires des particules quasiment impossible à réaliser. Cependant, il existe des études sur la dynamique des particules [Ozcan et al., 1999 ; Clerc et al., 2002 ; Van den Bergh, 2002 ; Trelea, 2003]. Ne pouvant appuyer ses choix sur des résultats théoriques, un concepteur de métaheuristique ne peut confirmer ou infirmer ses intuitions que par le biais de procédures empiriques.

Les résultats présentés dans la section 2.6 laissent apparaître que le défaut principal de TRIBES est le manque de diversité au sein de l'essaim. Les particules ont tendance à se "regrouper" dans

les vallées des optima locaux et à y rester. Aucune des trois stratégies de déplacement utilisées ne permet de disposer de mouvements assez larges pour "s'échapper" des vallées de ces optima locaux.

Dans le cadre de cette thèse, il a été introduit une nouvelle stratégie de déplacement qui autorise, dans certains cas, une particule à avoir un déplacement plus large que ne le peuvent proposer les trois autres stratégies de déplacement. On observe que, pour ces trois stratégies, chaque déplacement est influencé par la position courante de la particule, par son informateur cognitif et par son informateur social. Ainsi, si l'une ou plusieurs de ces positions sont piégées dans un optimum local, la particule va être attirée vers la "vallée" de cet optimum local, va "tourner" autour ce celui-ci (voir Figure 2.9) et, ainsi, il ne va pas lui être possible d'améliorer sa performance. L'idée de notre nouvelle stratégie de déplacement est de travailler non pas avec trois positions de référence, mais avec tout un ensemble de positions choisies soigneusement. La stratégie introduite va donc reposer sur une structure de type Algorithme à Estimation de Distribution (voir Section 1.2.2). Comme nous le détaillons dans l'Algorithme 6, on sélectionne un ensemble de particules considérées comme "intéressantes" dans l'essaim, on calcule le modèle probabiliste correspondant et, enfin, on tire aléatoirement la nouvelle position de la particule à l'aide de la distribution de probabilité ainsi calculée.

Le problème est de déterminer avec quel ensemble de positions la distribution de probabilité va être calculée. Pour un algorithme de type OEP comme TRIBES, le choix le plus simple et le plus naturel est de prendre comme famille de référence les $\overrightarrow{p_i},\ i \in \{1,\ldots,D\}$. L'utilisation d'un modèle de probabilité à support infini permet de garantir la possibilité d'un déplacement plus large pour la particule.

Dans notre cas, nous avons utilisé un modèle gaussien. Deux cas de figures sont alors envisageables, selon que les différentes variables sont indépendantes entre elles ou non. Si les variables sont indépendantes entre elles, les composantes du vecteur moyenne $\overrightarrow{\mu} = [\mu_1,\ldots,\mu_D]$ et les composantes du vecteur variance $\overrightarrow{\sigma} = [\sigma_1,\ldots,\sigma_D]$ sont calculées indépendamment à l'aide des équations (2.14) et (2.15). N est la taille de l'essaim et D est la dimension du problème.

$$\mu_i = \frac{1}{N}\sum_{j=1}^{N} p_{ji},\ i \in \{1,\ldots,D\} \tag{2.14}$$

$$\sigma_i^2 = \frac{1}{N-1}\sum_{j=1}^{N} \left(p_{ji}-\mu_i\right)^2,\ i \in \{1,\ldots,D\} \tag{2.15}$$

La $i^{ème}$ coordonnée de la nouvelle position est alors déterminée à l'aide d'un tirage aléatoire de la distribution gaussienne de moyenne μ_i et de variance σ_i.

Cependant, dans le cas d'applications réelles, il est courant que les variables soient dépendantes entre elles. Dans ce cas la loi gaussienne est définie par l'équation (2.16).

$$G\left(\overrightarrow{x}\right) = \frac{1}{(2\pi)^{\frac{D}{2}}\ |\Sigma|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}\left(\overrightarrow{x}-\overrightarrow{\mu}\right)^T \Sigma^{-1}\left(\overrightarrow{x}-\overrightarrow{\mu}\right)\right) \tag{2.16}$$

où $\overrightarrow{\mu} = [\mu_1,\ldots,\mu_D]$ est le vecteur moyenne et Σ est la matrice de covariance. La matrice de covariance peut être estimée à l'aide de l'équation (2.17), le vecteur de moyenne étant inchangé.

$$\hat{\Sigma} = \frac{1}{N-1}\sum_{j=1}^{N} \left(\overrightarrow{p_j}-\overrightarrow{\mu}\right)\left(\overrightarrow{p_j}-\overrightarrow{\mu}\right)^T \tag{2.17}$$

Dans ce cas, la nouvelle position est déterminée par tirage aléatoire de la distribution gaussienne de moyenne $\overrightarrow{\mu}$ et de matrice de covariance $\hat{\Sigma}$.

TABLE 2.4 – Comparaison entre algorithmes pour les fonctions unimodales 10D. Erreur moyenne et nombre d'évaluations de la fonction objectif moyen nécessaire pour atteindre un erreur inférieure à 10^{-6}.

	F1	F2	F3	F4	F5
Standard PSO 2006	0.00 (3375)	0.00 (8300)	7.12e+04 (100000)	0.00 (11562)	18.86 (67610)
EDA	0.00 (28732)	0.00 (28916)	0.00 (32922)	0.00 (32636)	233.50 (100000)
TRIBES	0.00 (1856)	0.00 (10452)	0.00 (10340)	0.00 (18966)	1.74 (60409)
TRIBES+	0.00 (1521)	0.00 (12011)	0.00 (9405)	0.00 (23783)	9.04e-04 (80832)

Bien évidemment, cette stratégie de déplacement ne doit pas être appliquée à toutes les particules. Cette stratégie de déplacement autorise des déplacements très larges et, donc, favorise un processus de diversification. De ce fait, cette stratégie doit être réservée aux particules ayant eu un mauvais comportement lors des dernières itérations. La stratégie de déplacement basée sur un EDA est donc attribuée aux particules ayant pour historique (- -), (- =) et (= -). Ainsi ces "mauvaises" particules vont pouvoir améliorer leur performances en allant explorer de nouvelles régions de l'espace de recherche.

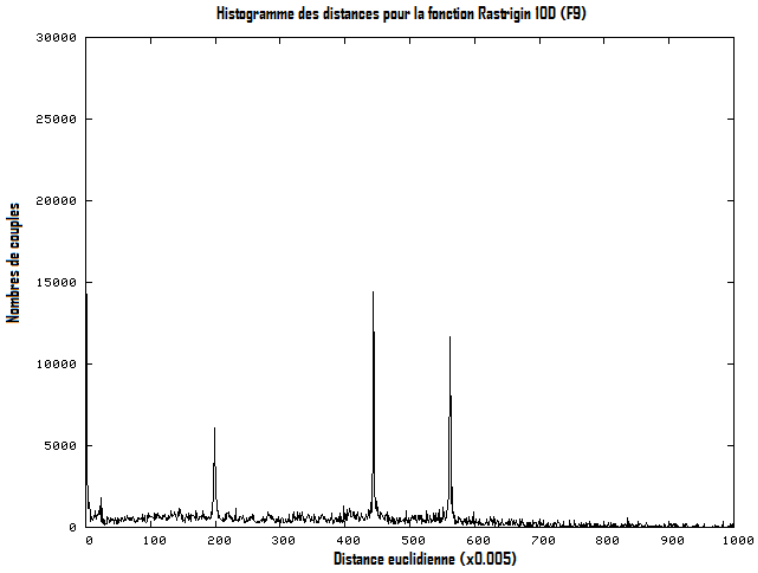
La Figure 2.14 montre l'histogramme des distances entre toutes les positions atteintes par les particules dans le cas de la fonction Rastrigin 10D. La Figure 2.14a est réalisée à l'aide de TRIBES classique et la Figure 2.14b est réalisée en introduisant la nouvelle stratégie de déplacement. Il apparaît que les positions atteintes dans le cas b) sont plus diverses que dans le cas classique. En effet, la Figure 2.14 est composée de quatre pics très fins. Cela implique qu'à la fin de l'exécution les particules ont principalement exploré quatre régions de l'espace de recherche. Les pics étant très étroits, on en déduit que les particules sont concentrées autour d'optima locaux. A l'opposé, sur la Figure 2.14b, les pics sont beaucoup plus larges. Ceci induit que, comme souhaité, on a exploré l'espace de recherche de manière plus large que dans le cas initial et, donc, on a amélioré le processus de diversification de TRIBES.

2.7.3 Expérimentation numérique des nouveaux modules de TRIBES

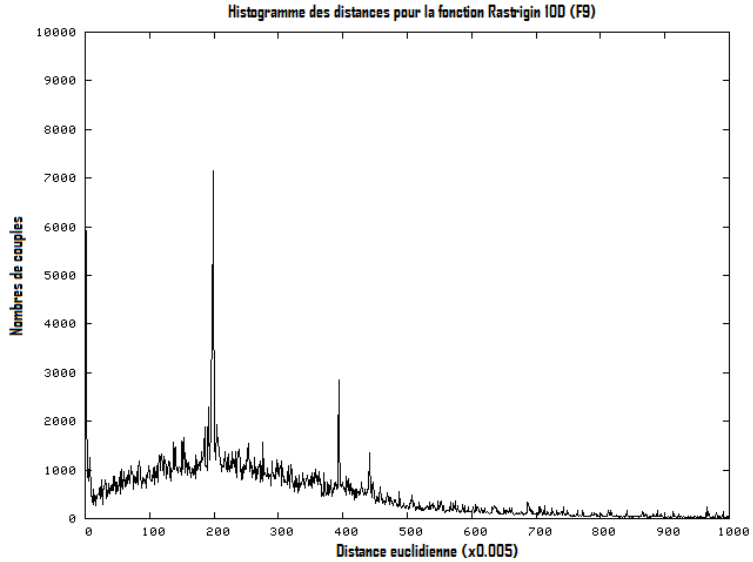
Dans tout ce paragraphe, TRIBES+ désignera l'algorithme TRIBES auquel on a ajouté l'initialisation régulière et la nouvelle stratégie de déplacement.

Dans cette section, la nouvelle version de TRIBES, i.e. avec une initialisation régulière et avec la nouvelle stratégie de déplacement, est évaluée sur les principales fonctions de test. Les différents résultats présentés sont obtenus pour des problèmes de dimension $D = 10$. La Table 2.4 présente les erreurs moyennes sur 25 exécutions des algorithmes et, entre parenthèses, le nombre moyen d'évaluations de la fonction objectif pour les fonctions unimodales F1 à F5. L'exécution de l'algorithme est stoppée si l'erreur est inférieure à 10^{-6} ou si le nombre d'évaluations de la fonction objectif dépasse $Max_{Fes} = 100000$. La Table 2.5 présente les erreurs moyennes sur 25 exécutions des algorithmes et, entre parenthèses, le nombre moyen d'évaluations de la fonction objectif pour les fonctions multimodales F6 à F11. L'exécution de l'algorithme est stoppée si l'erreur est inférieure à 10^{-2} ou si le nombre d'évaluations de la fonction objectif dépasse $Max_{Fes} = 100000$.

Les Tables 2.4 et 2.5 montrent que TRIBES+ donne des résultats compétitifs vis-à-vis des autres algorithmes de comparaison. Comparé au *Standard PS0 2006*, TRIBES+ donne des meilleurs résultats dans la grande majorité des cas. TRIBES+ est juste un peu plus lent pour les fonctions F4 et F5,



(a)



(b)

FIGURE 2.14 – Histogramme des distances entre particules pour la fonction Rastrigin 10D (F9). (a) TRIBES original, (b) TRIBES avec la nouvelle stratégie de déplacement.

TABLE 2.5 – Comparaison entre algorithmes pour les fonctions multimodales 10D. Erreur moyenne et nombre d'évaluations de la fonction objectif moyen nécessaire pour atteindre un erreur inférieure à 10^{-2}.

	F6	F7	F8	F9	F10	F11
Standard PSO 2006	1.88 (100000)	0.08 (100000)	20.11 (100000)	4.02 (100000)	10.18 (100000	4.72 (100000)
EDA	0.00 (43500)	0.53 (100000)	20.33 (100000)	32.28 (100000)	32.28 (100000)	8.27 (100000)
TRIBES	0.06 (100000)	0.07 (100000)	20.32 (100000)	8.55 (100000)	12.11 (100000)	5.68 (100000)
TRIBES+	0.12 (100000)	0.02 (100000)	20.35 (100000)	0.19 (100000)	4.12 (100000)	3.55 (100000)

ce qui peut s'expliquer par le fait que ces fonctions sont plus rapidement résolues par un algorithme disposant de beaucoup de particules au départ du processus (car elles sont relativement simples). On prouve donc ici que l'ajout des deux mécanismes présentés dans les sections précédentes apporte une réelle plus-value aux performances de TRIBES.

Ce travail a fait l'objet d'une publication [Cooren et al., 2007].

2.8 Conclusion

Dans ce chapitre, nous avons présenté TRIBES, un algorithme d'optimisation par essaim particulaire totalement adaptatif. TRIBES ne nécessite aucune intervention humaine pour la définition de ses paramètres et permet donc à l'utilisateur novice une prise en main plus facile de l'algorithme. Cependant, un tel algorithme ne peut être intéressant que si la facilité de prise en main ne se fait pas au détriment de la qualité des performances. Les tests proposés dans ce chapitre montrent que TRIBES donne des résultats compétitifs sur un très large panel de fonctions de test, et ce, même pour des dimensions élevées des problèmes. De même, au niveau de la rapidité de recherche de l'optimum global, TRIBES a des performances tout à fait correctes vis-à-vis des autres algorithmes, même si, dans certains cas, il est handicapé par le fait que son traitement ne débute qu'avec une seule particule.

Cependant, on constate que TRIBES souffre d'un manque de diversité au sein de son essaim. En effet, il apparaît dans de nombreux cas que la convergence de TRIBES est très rapide vers un optimum local et que les particules n'arrivent pas à "s'échapper" de ceux-ci. Cela entraîne une stagnation des particules et, donc, des performances mitigées.

Pour essayer d'apporter plus de diversité à l'essaim, nous avons introduit deux nouveaux mécanismes à TRIBES : une initialisation régulière et une nouvelle stratégie de déplacement. Le but de ces mécanismes est de permettre aux particules d'explorer l'espace de recherche le plus largement possible. Les résultats numériques montrent les effets bénéfiques de ces deux nouveaux mécanismes.

Chapitre 3

Élaboration de MO-TRIBES : algorithme d'optimisation multiobjectif sans paramètre

3.1 Introduction

Les métaheuristiques ont été à l'origine conçues pour traiter des problèmes mono-objectif, c'est-à-dire que la recherche de solutions pour un problème donné ne dépend que d'un seul critère. Or, l'expérience montre que de nombreux problèmes ne sont pas modélisables par un critère unique. En effet, pour une modélisastion correcte de ces problèmes, il ne faut pas définir un critère, mais plusieurs. La difficulté majeure lors de la résolution de ce type de problème provient du fait que les critères utilisés pour modéliser ces problèmes sont très souvent contradictoires. On ne recherche plus alors une solution unique au problème posé, mais plutôt un ensemble de solutions "de compromis" entre les différents critères, un expert dans le domaine traité ayant alors pour rôle de choisir parmi cet ensemble la solution finale. Dans le chapitre 1, nous avons décrit comment les principales métaheuristiques ont été modifiées et adaptées afin de pouvoir traiter des problèmes multiobjectifs. Il a été montré que les métaheuristiques multiobjectifs peuvent se diviser en trois classes : les méthodes agrégatives, les méthodes non-Pareto et les méthodes Pareto. Les méthodes basées sur la dominance de Pareto (cf. section 1.3.1) sont aujourd'hui les plus efficaces et, de ce fait, les plus utilisées.

Dans les chapitres 1 et 2, nous avons discuté la nécessité de réduire le nombre de paramètres de contrôle des algorithmes d'optimisation dans le cas mono-objectif. Nous avons conclu que, en vue d'un développement plus large de l'utilisation des métaheuristiques dans l'industrie, il est nécessaire de faciliter au maximum la prise en main des algorithmes par les utilisateurs novices. Parmi les objectifs identifiés pour réaliser cela, la réduction du nombre de paramètres de contrôle des algorithmes est l'un des plus importants. Ce constat, discuté dans le cas mono-objectif, est aussi valable dans le cas multiobjectif. En effet, les algorithmes multiobjectifs sont adaptés des méthodes mono-objectif donc ils sont, par nature, fortement dépendants de leurs paramètres. De plus, les mécanismes permettant de transformer un algorithme mono-objectif en algorithme multiobjectif introduisent très souvent la définition de nouveaux paramètres, donc une complexité supplémentaire lors de la prise en main de l'algorithme.

Le chapitre 2 a été consacré à la présentation de TRIBES, un algorithme d'Optimisation par Essaim Particulaire sans paramètres développé par Clerc. Le présent chapitre est consacré à la présentation de MO-TRIBES, une version multiobjectif de TRIBES, qui a été développée au cours de cette thèse. Comme TRIBES, MO-TRIBES présente la particularité de ne nécessiter le réglage

d'aucun paramètre de contrôle.

3.2 MO-TRIBES

MO-TRIBES a pour objectif d'être un algorithme d'optimisation par essaim particulaire multiobjectif sans paramètres de contrôle. Il reprend les principaux mécanismes de TRIBES auxquels sont ajoutés de nouveaux mécanismes destinés à traiter des problèmes multiobjectifs. MO-TRIBES a été conçu en utilisant une approche basée sur la dominance au sens de Pareto.

Dans le but d'adapter le paradigme de TRIBES à la résolution de problèmes multiobjectifs, il est clair que le schéma général de l'algorithme (cf. Algorithme 10) doit être modifié. En général, la résolution d'un problème multiobjectif pose trois problèmes [Ziztler et al., 2000] :

- trouver le plus d'éléments possibles appartenant au front de Pareto ;
- approcher le mieux possible le front de Pareto ;
- la répartition des solutions doit être la plus uniforme possible le long du front de Pareto approché.

Comme dans le cas des algorithmes évolutionnaires [Coello Coello et al., 2002], étendre l'algorithme d'OEP au traitement des problèmes multiobjectifs implique de répondre aux trois questions suivantes :

- Comment choisir les informatrices des particules de manière à favoriser les particules non-dominées ?
- Comment retenir les particules non-dominées pour que celles-ci soient non seulement non-dominées vis-à-vis de la population courante, mais aussi vis-à-vis des populations passées ?
- Comment maintenir la diversité dans l'essaim pour ne pas converger vers une solution unique ?

Les solutions non-dominées trouvées au cours de l'exécution sont stockées dans une *archive externe*, comme cela est déjà fait dans de nombreuses autres méthodes [Reyes-Sierra et al., 2006]. Ainsi, à la fin du traitement, l'archive contiendra toutes les solutions non-dominées rencontrées au cours de l'exécution et, de ce fait, l'approximation du front de Pareto du problème.

Les paragraphes suivants détaillent les modifications qui ont dû être apportées à TRIBES pour le transformer en MO-TRIBES.

3.2.1 Diversité au sein de l'essaim

Comme nous l'avons dit en introduction, l'algorithme ne doit pas converger vers une solution unique de l'espace de recherche. Il est donc primordial de maintenir une forte diversité au sein de l'essaim, si l'on veut obtenir une bonne approximation du front de Pareto.

Nous avons vu dans le chapitre précédent que, dans le cas mono-objectif, TRIBES converge très rapidement vers un optimum local, sans garantie que celui-ci soit l'optimum global. Un faible nombre d'itérations suffit donc pour que TRIBES atteigne un état de quasi-équilibre et, donc, que la diversité au sein de l'essaim ne soit plus maintenue.

Dans MO-TRIBES, ce défaut inhérent au paradigme de l'OEP est utilisé pour maintenir la diversité au sein de l'essaim. En effet, considérant que l'essaim va converger au bout d'un faible nombre d'itérations vers un état de quasi-équilibre, on peut considérer qu'il n'est pas utile de continuer la recherche à partir de ce moment là, et qu'il est alors préférable de recommencer une nouvelle recherche. En d'autres termes, si, après un certain nombre d'itérations, les particules n'ont plus de

Algorithme 11 Mise à jour de la taille de l'archive

Si $reinit = 0$
  $archiveSize = e^k$
Sinon
  $archiveSize = archiveSize + 10 \cdot \ln(1 + nDomPrev)$
Fin Si

déplacements significatifs, on peut considérer qu'il est préférable de ré-initialiser l'essaim, plutôt que de continuer la recherche avec l'essaim courant. De ce fait, de nouvelles zones de l'espace de recherche vont pouvoir être explorées.

En pratique, notre objectif est de trouver des solutions non-dominées. On considère donc qu'un essaim a atteint un état de quasi-équilibre s'il n'arrive plus à trouver de nouvelles solutions non-dominées. L'essaim sera donc réinitialisé si, entre deux itérations successives de MO-TRIBES, l'algorithme n'a pas trouvé de nouvelles solutions non-dominées. On assure ainsi un renouvellement assez fréquent de la population et, donc, on multiplie nos chances de trouver de nouvelles solutions non-dominées.

3.2.2 Techniques d'archivage

L'archive externe contient les solutions non-dominées trouvées tout au long du processus d'optimisation. On a vu dans le paragraphe 1.3.5 que, pour éviter une explosion de la complexité de l'algorithme, il est nécessaire de borner la taille de l'archive. Borner la taille de l'archive permet de gagner en complexité et, donc, en temps, mais introduit un nouveau paramètre, la taille de l'archive, ainsi qu'une nouvelle règle, qui permet de gérer la mise à jour de l'archive. En effet, lorsque l'archive est pleine, il est nécessaire de disposer d'une règle qui définisse les conditions dans lesquelles les nouvelles solutions non-dominées remplacent celles déjà présentes dans l'archive.

Comme MO-TRIBES a pour but d'être un algorithme sans paramètres de contrôle, il nous est imposé de trouver une règle qui permette d'adapter la taille de l'archive en fonction du comportement de l'algorithme. La règle proposée a été définie empiriquement lors de la conception de MO-TRIBES. Tout d'abord, on admet que la taille de l'archive est dépendante du nombre d'objectifs. Plus le nombre d'objectifs est important, plus il est nécessaire de disposer de beaucoup de solutions non-dominées pour approcher le front de Pareto. Ensuite, à chaque nouvelle adaptation, la taille de l'archive est mise à jour, en fonction du nombre de solutions non-dominées trouvées entre les deux adaptations.

Cette procédure est résumée par l'Algorithme 11. k est le nombre de fonctions objectifs, *nDomPrev* est le nombre de solutions non-dominées trouvées depuis la dernière mise à jour de la taille de l'archive et *reinit* est le nombre de ré-initialisations de l'essaim, depuis le début de traitement.

La diversité des solutions non-dominées stockées au sein de l'archive est maintenue à l'aide d'un critère basé sur la *distance d'encombrement* [Deb et al., 2002]. La *distance d'encombrement* du $i^{ème}$ élément de l'archive approche le volume du plus grand hypercube contenant la $i^{ème}$ solution de l'archive sans en inclure d'autres. L'idée est de maximiser la *distance d'encombrement* entre les solutions stockées dans l'archive, afin d'obtenir une répartition des particules la plus uniforme possible le long du front de Pareto. Sur la Figure 3.1, la *distance d'encombrement* de la solution i est la largeur moyenne de l'hypercube matérialisé en pointillé.

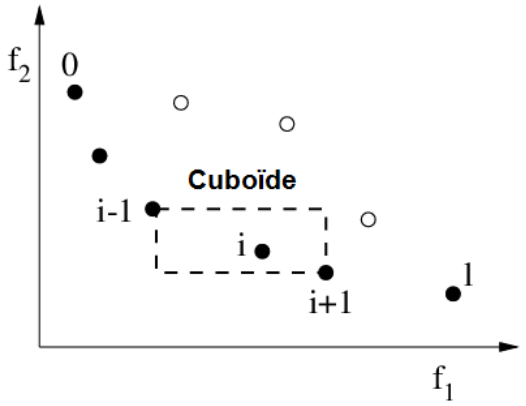


FIGURE 3.1 – *Distance d'encombrement.*

Algorithme 12 Mise à jour de l'archive

Pour i=1 à *tribeNb*
  Pour j=1 à *tribe[i].explorerNb*
    Si *tribe[i].particle[j]* domine des éléments de l'archive
      Effacer les éléments dominés
    Fin si
    Si t*ribe[i].particle[j]* est non-dominé
      Si *archiveSize* $\neq$ *nonDomCtr*
        Ajouter t*ribe[i].particle[j]* à l'archive
      Sinon
        Calculer les *distances d'encombrement* de chaque solution
        Trier les éléments de l'archive en fonction de la *distance d'encombrement*
        Remplacer la solution avec la plus faible *distance d'encombrement* par *tribe[i].particle[j]*
      Fin si
    Fin si
  Fin pour
Fin pour

Dans ce qui suit, la norme utilisée pour la *distance d'encombrement* est la norme 2. Dans les cas réels, il est possible que la norme 2 ne soit pas applicable pour certains objectifs, ce qui oblige à utiliser une norme appropriée pour la définition de la *distance d'encombrement*. Ce cas n'a pas été exploré dans cette thèse.

L'archive est mise à jour, suivant la procédure décrite par l'Algorithme 12. *tribeNb* est le nombre de tribus dans l'essaim, *tribe[i].explorerNb* est le nombre de particules à l'intérieur de la $i^{ème}$ tribu, *tribe[i].particle[j]* est la $j^{ème}$ particule de la $i^{ème}$ tribu, *nonDomCtr* est le nombre de solutions non-dominées stockées dans l'archive et *archiveSize* est la taille maximale de l'archive.

3.2.3 Choix des informateurs

La solution à un problème d'optimisation multiobjectif est un ensemble de solutions non-dominées équivalentes. Il n'y a aucune solution qui puisse être dite "meilleure" que les autres. Le concept d'informatrice dans MO-TRIBES est, de ce fait, différent de celui de TRIBES.

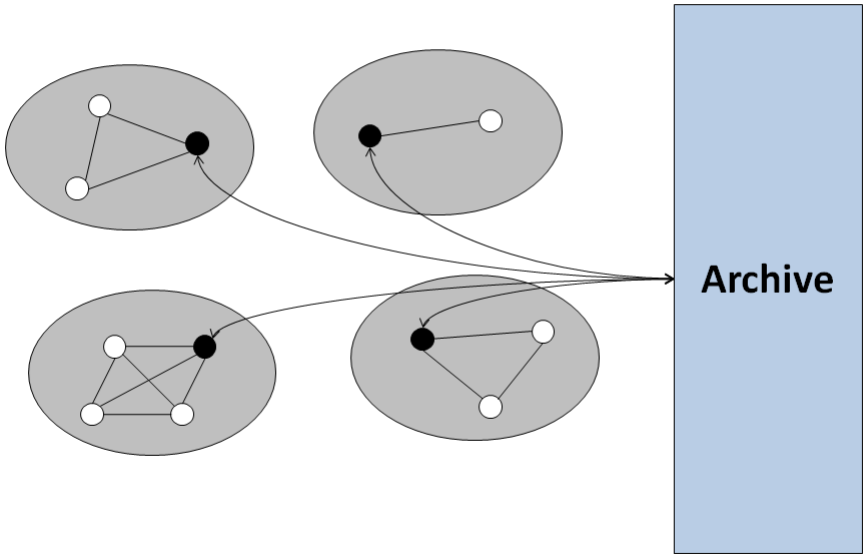


FIGURE 3.2 – Topologie des liens d'information dans MO-TRIBES.

Pour une particule donnée $\overrightarrow{X}$, le premier informateur est $\overrightarrow{p}$, l'informateur cognitif. $\overrightarrow{p}$ modélise la tendance de la particule à suivre sa propre expérience. A l'itération t, $\overrightarrow{p}$ est mis à jour si, et seulement si, la nouvelle position $\overrightarrow{X}$ domine $\overrightarrow{p}$. Le critère de dominance est donc utilisé pour mettre à jour l'informateur cognitif. Le second informateur à déterminer est $\overrightarrow{g}$, l'informateur social. $\overrightarrow{g}$ modélise l'influence de l'essaim sur le comportement de la particule. Le choix de $\overrightarrow{g}$ est dépendant du statut de $\overrightarrow{X}$ au sein de la tribu. Si $\overrightarrow{X}$ est une particule quelconque de la tribu, $\overrightarrow{g}$ sera la meilleure position jamais atteinte par le *chaman* de la tribu. Si $\overrightarrow{X}$ est le chaman de la tribu, $\overrightarrow{g}$ est choisi aléatoirement au sein de l'archive, afin de donner une préférence aux solutions non-dominées. Choisir aléatoirement l'informateur social du chaman au sein de l'archive permet d'assurer une certaine diversité au sein de l'essaim. La figure 3.2 illustre la topologie des liens d'information. Les particules noires sont les chamans des différentes tribus.

L'Algorithme 13 résume la mise à jour des liens d'information à l'instant t. Les éléments de cet algorithme sont les suivants :

- *tribe[i].particle[j]* est la $j^{ème}$ particule de la $i^{ème}$ tribu ;
- *tribe[i].particle[j].p* est l'informateur cognitif de *tribe[i].particle[j]* ;
- *tribe[i].particle[j].g* est l'informateur social de *tribe[i].particle[j]* ;
- *tribe[i].particle[chaman]* est le *chaman* de la $i^{ème}$ tribu ;
- *U(archive)* est une solution uniformément choisie dans l'archive.

Algorithme 13 Processus d'information

Pour i=1 à *tribeNb*
  Pour j=1 à *tribe[i].explorerNb*
    Si *tribe[i].particle[j]* domine *tribe[i].particle[j].p*
      tribe[i].particle[j].p=tribe[i].particle[j]
    Fin si
    Si *tribe[i].particle[j]* est le chaman de la tribu i
      tribe[i].particle[j].g=U(archive)
    Sinon
      tribe[i].particle[j].g=tribe[i].particle[chaman].p
    Fin si
  Fin pour
Fin pour

3.2.4 Stratégies de déplacement

Les stratégies de déplacement sont identiques à celles de TRIBES. Elles sont résumées dans la Table 3.1. La fonction λ est définie par l'équation (3.1).

$$\lambda\left(\overrightarrow{x}\right) = \frac{1}{k}\sum_{i=1}^{k}\frac{f_i\left(\overrightarrow{x}\right)}{\max_{j\in\{1,\ldots,k\}}\left(f_j\left(\overrightarrow{x}\right)\right)} \tag{3.1}$$

3.2.5 Algorithme

L'Algorithme 14 donne un pseudo-code de MO-TRIBES. *NL* est le nombre de liens d'informations lors de la précédente adaptation de l'essaim et n est le nombre d'itérations effectuées depuis cette dernière adaptation.

3.3 Résultats numériques de MO-TRIBES

3.3.1 Fonctions de test

De nombreuses fonctions de test ont été proposées pour évaluer les algorithmes multiobjectifs [Fonseca et al., 1996 ; Laumanns et al., 2001 ; Kursawe, 1990]. Ces problèmes ont été conçus pour présenter les caractéristiques majeures que peuvent présenter les fronts de Pareto (concavité/convexité, continuité/discontinuité, séparabilité, modalité, etc...). Les fonctions utilisées pour évaluer MO-TRIBES sont présentées dans la Table A.1.

3.3.2 Critères de performance

Dans le cas multiobjectif, les algorithmes ne peuvent plus être jugés sur la qualité de la solution fournie. De ce fait, de nombreuses métriques ont été définies, afin de juger de la qualité d'un front de Pareto [Okabe et al., 2003]. Dans la suite de ce paragraphe, nous présentons les trois métriques utilisées pour évaluer MO-TRIBES.

TABLE 3.1 – Stratégies de déplacement.

Historique de la particule	Stratégie de déplacement	Equation
(= +) (+ +)	locale par gaussiennes indépendantes	$X_j = g_j + alea_{normal}\left(g_j - X_j, \|g_j - X_j\|\right),\ j \in \{1, \ldots, D\}$
(+ =) (- +)	pivot bruité	$\overrightarrow{X} = c_1 \cdot alea_{sphere}\left(H_p\right) + c_2 \cdot alea_{sphere}\left(H_g\right)$ $c_1 = \frac{\lambda(\overrightarrow{p})}{\lambda(\overrightarrow{p})+\lambda(\overrightarrow{g})}$ $c_2 = \frac{\lambda(\overrightarrow{g})}{\lambda(\overrightarrow{p})+\lambda(\overrightarrow{g})}$
(- -) (= -) (+ -) (- =) (= =)	pivot	$\overrightarrow{X} = c_1 \cdot alea_{sphere}\left(H_p\right) + c_2 \cdot alea_{sphere}\left(H_g\right)$ $b = N\left(0,\ \frac{\lambda(\overrightarrow{p})-\lambda(\overrightarrow{g})}{\lambda(\overrightarrow{p})+\lambda(\overrightarrow{g})}\right)$ $\overrightarrow{X} = (1+b) \cdot \overrightarrow{X}$

Algorithme 14 Algorithme de MO-TRIBES.

Initialiser la taille de l'archive
Initialiser un essaim de particules avec des positions aléatoires
Pour chaque particule i, $\overrightarrow{p_i}$ est initialisé à $\overrightarrow{X_i}$
Évaluer les fonctions objectifs pour chaque particule et initialiser $\overrightarrow{g_i}$
Insérer les particules non-dominées dans l'archive
Tant que le critère d'arrêt n'est pas atteint
  Choisir les stratégies de déplacement
  Mettre à jour les vitesses et les positions des particules
  Évaluer les fonctions objectifs pour chaque particule
  Mettre à jour $\overrightarrow{p_i}$ et $\overrightarrow{g_i}$
  Mettre à jour l'archive
  Si $n<NL$
    Adaptations structurelles (ajout/élimination de particules, mise à jour du réseau d'information, réinitialisation de l'essaim...)
    Si $nDomPrev=0$
      Redémarrer l'algorithme
      Mettre à jour la taille de l'archive
    Fin Si
    Calculer NL
  Fin Si
Fin Tant que

Covariance : La *covariance* $C(U,V)$ mesure le ratio de solutions de l'archive V qui sont dominées par des solutions de l'archive U. $C(U,V)$ est défini par l'équation (3.2) :

$$C(U,V) = \frac{|\{b \in V \mid \exists a \in U,\ a \text{ domine } b\}|}{|V|} \tag{3.2}$$

où l'opérateur $|.|$ donne le cardinal de l'ensemble.

$C(U,V) = 1$ implique que tous les vecteurs contenus dans V sont dominés par ceux contenus dans U. A l'opposé, $C(U,V)$ est égal à 0 quand aucun des vecteurs de V n'est dominé par un des vecteurs de U. Ainsi, si U et V sont deux fronts de Pareto donnés par deux algorithmes différents A_U et A_V, plus le couple $(C(U,V), C(V,U))$ est proche de $(1,0)$, plus l'algorithme A_U aura tendance à être meilleur que l'algorithme A_V. Deux algorithmes ont des performances semblables si $(C(U,V), C(V,U))$ tend vers $(0.5, 0.5)$.

Espacement : L'*espacement* S mesure la distribution des solutions stockées dans l'archive le long du front de Pareto. $S(A)$ est défini par les équations (3.3) et (3.4) :

$$S(A) = \sqrt{\frac{1}{|A|-1} \cdot \sum_{i=1}^{|A|} (d_i - d_{moy})^2} \tag{3.3}$$

$$d_i = \min_{s_j \in A \wedge s_j \neq s_i} \left(\sum_{u=1}^{k} |f_u(s_i) - f_u(s_j)| \right) \tag{3.4}$$

où A est l'archive considérée, $|A|$ est le nombre de solutions stockées dans l'archive, k est le nombre de fonctions objectifs et d_{moy} est la distance moyenne. Plus $S(A)$ est grand, mieux les solutions sont réparties le long du front de Pareto.

Largeur de front : La *largeur de front* MS mesure la largeur du front de Pareto. MS est défini par l'équation (3.5) :

$$MS = \sqrt{\sum_{u=1}^{k} \left(\max_{i=1}^{|A|} (f_u(s_i)) - \min_{i=1}^{|A|} (f_u(s_i)) \right)^2} \tag{3.5}$$

3.3.3 Résultats expérimentaux de MO-TRIBES

Les résultats obtenus avec MO-TRIBES sont comparés à NSGA-II [Deb et al., 2002], qui est connu comme le meilleur algorithme d'optimisation multiobjectif, et à MOPSO [Raquel et al., 2005], qui est l'un des meilleurs algorithmes d'OEP multiobjectif. NSGA-II est exécuté avec une population de taille 100, une probabilité de croisement de 0.8 et une probabilité de mutation de 0.1. MOPSO est exécuté avec une population de taille 100, un coefficient d'inertie w égal à 0.4 et des coefficients d'accélération c_1 et c_2 égaux à 1. Pour ces deux algorithmes, la taille maximale de l'archive externe est égale à 100. MO-TRIBES n'a aucun paramètre de contrôle à définir. Dans le code source de MO-TRIBES, la taille maximale de l'archive est cependant bornée à 100. Cette borne n'est ici insérée que par prévention, la taille de l'archive étant bien régulée par les équations de l'Algorithme 11 et ne dépassant jamais 100. Le critère d'arrêt utilisé pour chacun des algorithmes est $Max_{Fes} = 50000$.

TABLE 3.2 – Comparaison des algorithmes basée sur la *covariance*.

	Deb	ZDT1	ZDT2	ZDT3	ZDT6	MOP5	MOP6
NSGA-II	(0.14, 0.18)	(0.00, 0.85)	(0.00, 0.35)	(0.00,0.76)	(1.00, 0.00)	(0.52, 0.07)	(0.03, 0.03)
MOPSO	(0.01, 0.27)	(1.00, 0.00)	(0.71, 0.05)	(1.00, 0.00)	(1.00, 0.00)	(0.40, 0.03)	(0.06, 0.01)

TABLE 3.3 – Comparaison des algorithmes basée sur l'*espacement*.

	Deb	ZDT1	ZDT2	ZDT3	ZDT6	MOP5	MOP6
NSGA-II	0.0313	0.0096	0.0089	0.0098	0.0067	0.3515	0.0076
MOPSO	0.0236	0.0072	0.0070	0.0084	0.1567	0.2839	0.0059
MO-TRIBES	0.0188	0.0047	0.0130	0.0336	0.0870	0.2237	0.0040

La Table 3.2 présente les couples moyens $(C(A_{MO-TRIBES}, C_X), C(A_X, A_{MO-TRIBES}))$ sur 25 exécutions des algorithmes. X symbolise soit NSGA-II soit MOPSO.

La Table 3.3 présente les *espacements* moyens sur 25 exécutions des algorithmes NSGA-II, MOPSO et MO-TRIBES.

La Table 3.4 présente les *largeurs de front* moyennes sur 25 exécutions des algorithmes NSGA-II, MOPSO et MO-TRIBES.

Les Figures 3.3 à 3.9 représentent une comparaison entre les fronts de Pareto fournis par NSGA-II et MO-TRIBES. Les fronts matérialisés par des points sont ceux obtenus à l'aide de NSGA-II, ceux matérialisés par des croix étant obtenus à l'aide de MO-TRIBES.

3.3.4 Discussion

La Table 3.3 montre que MO-TRIBES présente des résultats compétitifs vis-à-vis de ceux de NSGA-II. Pour la fonction ZDT6, toutes les solutions non-dominées trouvées par NSGA-II sont dominées par au moins une des solutions non-dominées trouvées par MO-TRIBES. MO-TRIBES est aussi nettement plus performant pour le problème MOP5. Pour les problèmes ZDT1, ZDT2 et ZDT3, MO-TRIBES ne semble pas encore atteindre les mêmes performances que NSGA-II. Les Figures 3.4 à 3.6 montrent que l'on n'est pas trop éloigné des résultats optimaux pour ZDT1 et ZDT3 alors que, dans le cas de ZDT2, MO-TRIBES semble ne pas arriver à correctement approcher le front de Pareto. Ce manque d'efficacité provient principalement du fait que MO-TRIBES ne démarre qu'avec une seule particule, contre 100 à NSGA-II, ce qui peut amener le traitement du problème par MO-TRIBES à être un peu plus long. Pour les problèmes Deb et MOP6, les résultats sont sensiblement

TABLE 3.4 – Comparaison des algorithmes basée sur la *largeur de front*.

	Deb	ZDT1	ZDT2	ZDT3	ZDT6	MOP5	MOP6
NSGA-II	6.41	1.41	1.41	1.96	1.12	83.35	1.69
MOPSO	6.41	1.43	1.40	2.02	8.34	62.78	1.69
MO-TRIBES	6.41	1.43	1.41	2.05	1.22	102.33	1.69

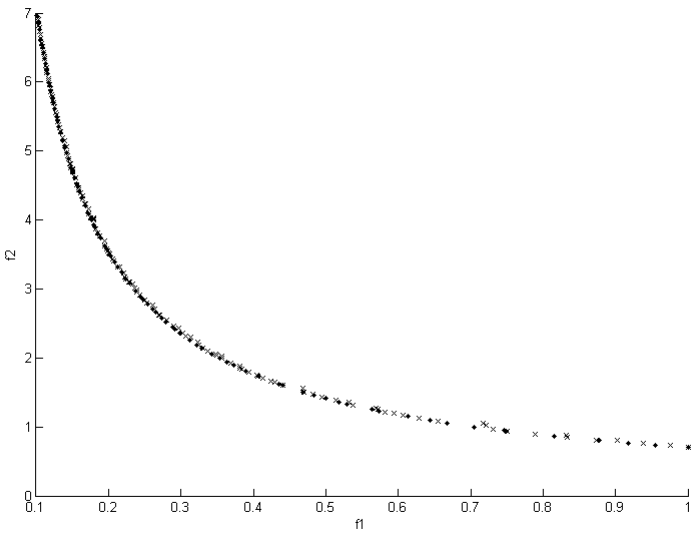


FIGURE 3.3 – Fronts de Pareto de NSGA-II et MO-TRIBES pour la fonction Deb. (×) Front de Pareto donné par MO-TRIBES, (.) Front de Pareto donné par NSGA-II.

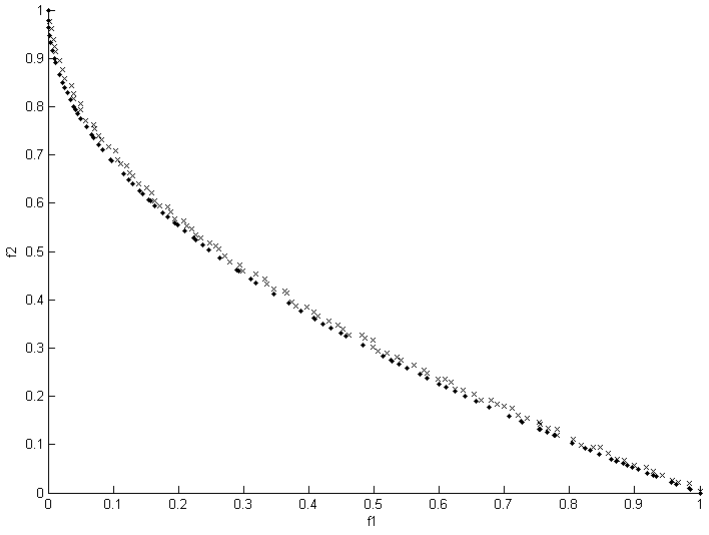


FIGURE 3.4 – Fronts de Pareto de NSGA-II et MO-TRIBES pour la fonction ZDT1. (×) Front de Pareto donné par MO-TRIBES, (.) Front de Pareto donné par NSGA-II.

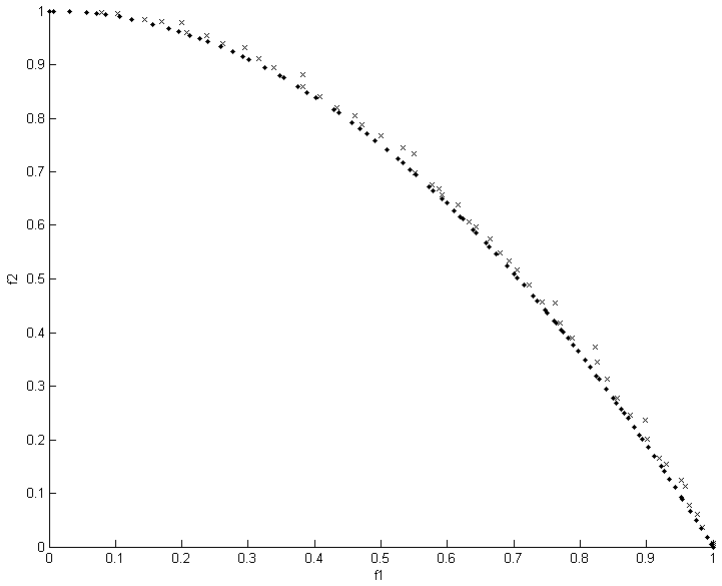


FIGURE 3.5 – Fronts de Pareto de NSGA-II et MO-TRIBES pour la fonction ZDT2. (×) Front de Pareto donné par MO-TRIBES, (.) Front de Pareto donné par NSGA-II.

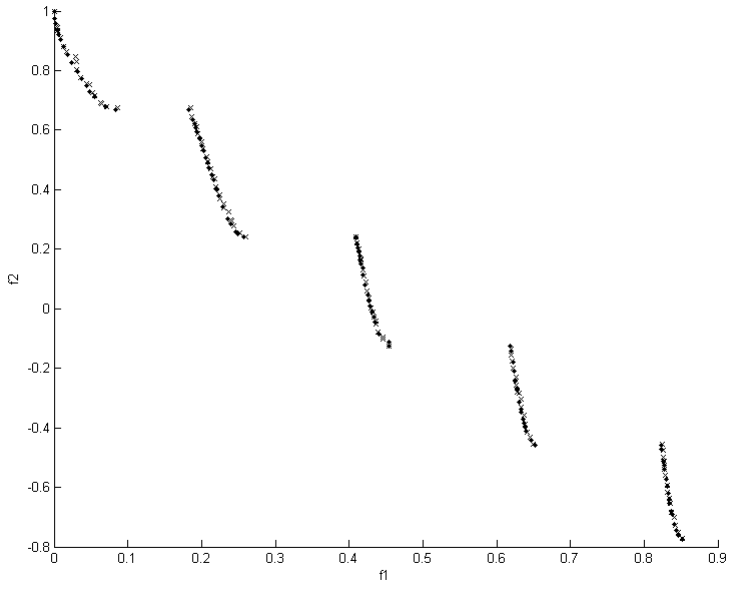


FIGURE 3.6 – Fronts de Pareto de NSGA-II et MO-TRIBES pour la fonction ZDT3. (×) Front de Pareto donné par MO-TRIBES, (.) Front de Pareto donné par NSGA-II.

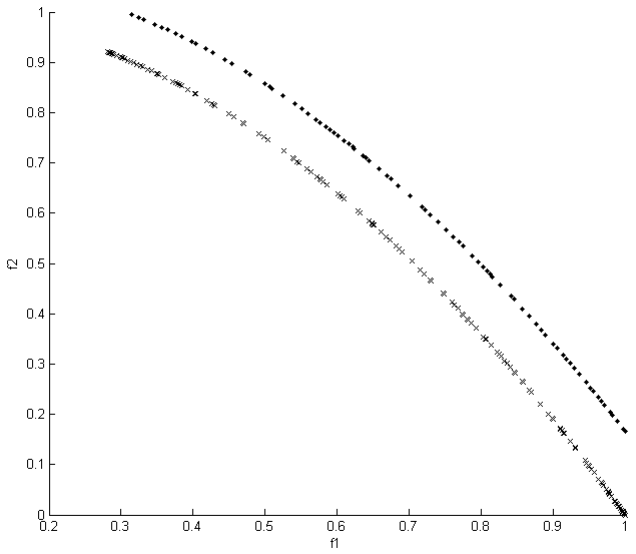


FIGURE 3.7 – Fronts de Pareto de NSGA-II et MO-TRIBES pour la fonction ZDT6. (×) Front de Pareto donné par MO-TRIBES, (.) Front de Pareto donné par NSGA-II.

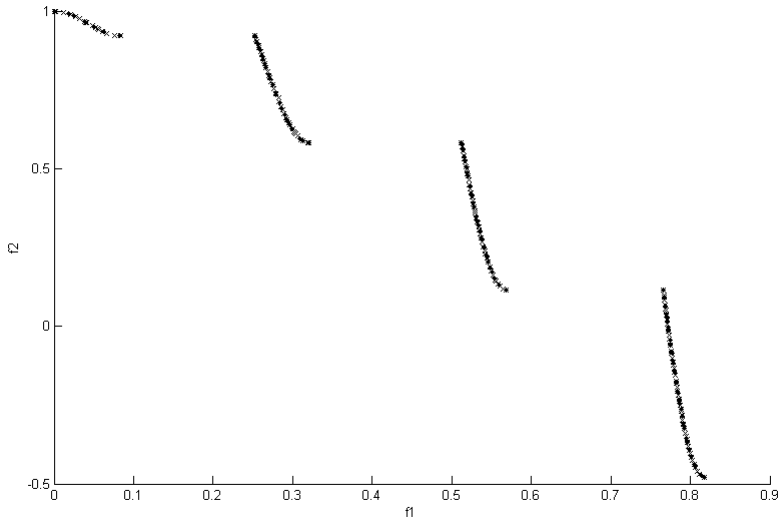


FIGURE 3.8 – Fronts de Pareto de NSGA-II et MO-TRIBES pour la fonction MOP6. (×) Front de Pareto donné par MO-TRIBES, (.) Front de Pareto donné par NSGA-II.

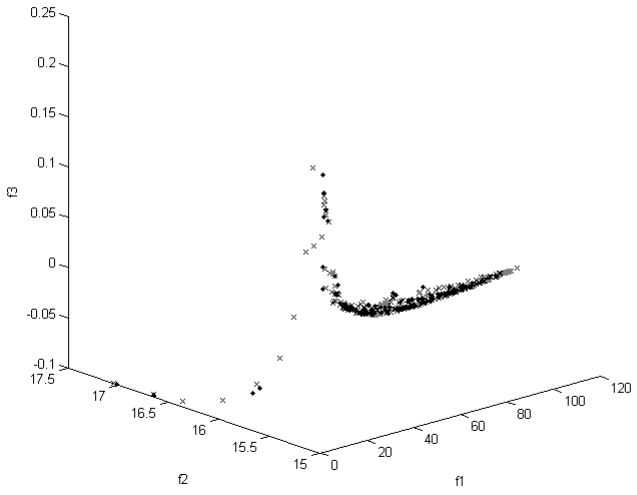


FIGURE 3.9 – Fronts de Pareto de NSGA-II et MO-TRIBES pour la fonction MOP5. (×) Front de Pareto donné par MO-TRIBES, (.) Front de Pareto donné par NSGA-II.

équivalents.

La Table 3.2 montre que MO-TRIBES donne de bien meilleurs résultats que MOPSO. Les résultats obtenus à l'aide de MOPSO sont meilleurs que ceux de MO-TRIBES seulement pour la fonction Deb. Ils sont similaires dans le cas du problème MOP6 et à l'avantage de MO-TRIBES pour tous les autres problèmes.

La Table 3.3 nous informe que les solutions non-dominées trouvées par MO-TRIBES sont assez bien réparties le long du front de Pareto. De même, la Table 3.4 montre que MO-TRIBES fournit des fronts de Pareto de même largeur que ceux fournis par MOPSO et NSGA-II.

Toutes les données numériques observées dans les Tables 3.2 à 3.4 sont confirmées par les impressions visuelles données par les Figures 3.3 à 3.9. On peut noter que, pour le problème ZDT1, les solutions non-dominées trouvées par MO-TRIBES sont mieux réparties sur la partie droite du front que celles données par NSGA-II. De même, pour le problème MOP5, on arrive à trouver davantage de solutions non-dominées dans le plan $f_2 - f_3$ avec MO-TRIBES qu'avec TRIBES.

3.4 Conclusion

Dans ce chapitre, nous avons présenté MO-TRIBES, un algorithme adaptatif d'optimisation par essaim particulaire sans paramètres de contrôle. Comme nous l'avons déjà observé dans le cas mono-objectif, l'utilisation d'algorithmes adaptatifs permet aux utilisateurs de métaheuristiques de s'affranchir du problème de définition des paramètres de l'algorithme, ceux-ci étant automatiquement calculés au cours de l'exécution de l'algorithme.

MO-TRIBES est une OEP multiobjectif disposant d'une archive externe dans laquelle sont stockées les solutions non-dominées trouvées tout au long du traitement. La taille de l'archive est déterminée de manière adaptative et la diversité des solutions acceptées au sein de l'archive est gérée grâce à un critère basé sur la maximisation de la *distance d'encombrement* des solutions de l'archive. De plus, la diversité est aussi assurée au sein de l'essaim de particules à l'aide d'une stratégie de

"redémarrage" de l'algorithme. Les comparaisons avec deux algorithmes très populaires montrent que, même si son comportement n'est pas "guidé" par des paramètres soigneusement choisis par l'utilisateur, MO-TRIBES donne de bonnes approximations des fronts de Pareto, pour lesquelles les solutions sont bien réparties le long du front.

Pour résumer, on peut dire que MO-TRIBES est un algorithme compétitif pour résoudre des problèmes d'optimisation multiobjectif, ce qui évite aux non-spécialistes d'avoir à définir des paramètres de contrôle.

Ce travail a fait l'objet d'une soumission d'article [Cooren et al., 2008b].

Chapitre 4

Application de TRIBES en segmentation d'images médicales

4.1 Introduction

L'imagerie est un domaine qui est devenu prépondérant dans de nombreux domaines, tels que la surveillance, la médecine ou encore la prospection pétrolière. Les images pouvant apporter des informations très diverses, il est devenu nécessaire de disposer d'algorithmes de traitement automatique des images. Un système de traitement automatique d'images est composé de deux niveaux : le premier est consacré au traitement numérique des images, au sens large, avec des opérations telles que le codage, l'amélioration ou encore la restauration des images, le deuxième est dédié à l'analyse symbolique des images, dans le but d'extraire l'information contenue dans l'image.

La segmentation occupe une place importante, car elle est située à l'articulation entre le traitement et l'analyse des images. La complexité du problème, la diversité des images et l'évaluation assez empirique des résultats en font un axe de recherche très actif. Le but de la segmentation est de partitionner une image en plusieurs régions homogènes, au sens d'un critère fixé a priori. On passe ainsi d'une information de type "intensité lumineuse" à une information de type "appartenance du pixel à une région donnée". D'un point de vue purement pratique, l'opération de segmentation revient à donner à chaque pixel de l'image un label d'appartenance à une région donnée. La partition de l'image en régions homogènes permet de fournir des données simplifiées qui vont permettre de faciliter la tâche d'un système de reconnaissance ou d'un système de classification.

De très nombreuses méthodes de segmentation existent. Ce chapitre traite du problème de segmentation par seuillage d'images. Le seuillage d'image est une méthode de segmentation supervisée, c'est-à-dire que le nombre de régions et leurs propriétés sont fournis au préalable par l'utilisateur. La segmentation est effectuée en déterminant, pour chaque pixel, la classe dont les propriétés se rapprochent le plus de celles observées en ce site. La technique de seuillage est basée sur l'hypothèse que les différentes régions de l'image peuvent être différenciées par leurs niveaux de gris. Cette méthode repose donc sur l'utilisation de l'histogramme de l'image traitée. Le seuillage de l'image en N classes revient donc à trouver les N-1 seuils qui vont partitionner l'histogramme en N zones. Le problème de seuillage est défini dans la première section de ce chapitre.

Dans ce chapitre, le problème de segmentation sera abordé selon deux critères différents. Tout d'abord, nous présentons un problème de segmentation d'images IRM de cerveau. Le critère de segmentation utilisé dans ce cas est l'entropie exponentielle 2D. Les seuils optimaux sont obtenus lorsque l'entropie exponentielle 2D est maximale. Ce problème est résolu à l'aide de TRIBES. Ensuite, nous montrons comment on peut faciliter le processus de segmentation en approchant l'histogramme

par une somme de gaussiennes. Approcher l'histogramme par une somme de gaussiennes permet de disposer d'une expression analytique de l'histogramme et, donc, de calculer plus facilement les seuils optimaux.

Cette application ayant été réalisée en cours de thèse, elle permet de corroborer les observations faites sur le comportement de TRIBES et, justifie donc les recherches d'améliorations pour TRIBES initiées avec TRIBES+.

4.2 Problème de seuillage

4.2.1 Définition du problème

Le problème de seuillage consiste à diviser l'image en N régions non forcément connexes, N étant défini au préalable par l'utilisateur. Le processus de segmentation est uniquement basé sur l'hypothèse que les différentes régions de l'espace sont différenciables par leurs niveaux de gris. Segmenter l'image en N régions revient donc à trouver *N-1* seuils qui vont diviser l'histogramme en N régions.

Soit f l'image à segmenter, l'image segmentée g est définie par :

$$g(x,y) = k \; si \; T_k \leq f(x,y) < T_{k+1}, \; k \in \{0, \ldots, N-1\} \tag{4.1}$$

où x et y sont les coordonnées du pixel courant, N est le nombre de classes et $T_0, \ldots, T_N$ sont les bornes des différentes régions en niveaux de gris. Le but d'une méthode de seuillage est de trouver les seuils optimaux $T_1, \ldots, T_{N-1}$, T_0 et T_N étant les bornes de l'ensemble des niveaux de gris possibles.

Le seuillage manuel d'image se déroule en quatre étapes :

- observation de l'histogramme ;
- choix des seuils dans les vallées de l'histogramme ;
- définition des classes de régions par intervalles de couleurs ;
- classement des pixels.

Le but d'un algorithme de seuillage est de proposer une méthode automatique qui permette à l'utilisateur de s'affranchir des quatre étapes précédemment citées.

4.2.2 État de l'art des méthodes de seuillage

Les méthodes de seuillage peuvent être divisées en deux classes : les méthodes *paramétriques*, qui sont basées sur l'hypothèse que les densités de probabilité des niveaux de gris des différentes classes sont gaussiennes, et les méthodes *non-paramétriques*, qui reposent sur l'optimisation d'un ou plusieurs critères a posteriori.

4.2.2.1 Méthodes paramétriques de seuillage

Les méthodes paramétriques de seuillage supposent que les différentes classes de l'image suivent une densité de probabilité pré-définie. L'histogramme est approché par une somme de densités de probabilité et les seuils sont choisis aux intersections de celles-ci. Le problème se décompose ici en deux étapes : estimer les paramètres des densités de probabilité et déterminer les seuils de segmentation. La problématique de ce genre de méthode est définie en détail dans la section 4.4. La plupart des méthodes existantes sont présentées dans [Sezgin et al., 2004 ; Sahoo et al., 1998 ; Gonzalez et al., 2002].

4.2.2.2 Méthodes non-paramétriques de seuillage

Les deux méthodes de référence dans cette catégorie sont la méthode d'Otsu [Otsu, 1979] et la méthode de Kapur et al. [Kapur et al., 1985]. La plupart des techniques parues ensuite sont basées sur ces deux principes.

4.2.2.2.1 Méthode d'Otsu Otsu décrit trois critères principaux de discrimination des classes : la variance interclasse, la variance intraclasse et la variance totale. Les trois critères sont équivalents et, suivant la situation, l'un ou l'autre peut être choisi. Cependant, du fait de sa simplicité de mise en oeuvre, la variance interclasse est souvent choisie.

Le calcul des critères est basé sur la densité de probabilité des différents pixels de l'image. Si l'on note h l'histogramme de l'image, celle-ci est donnée par :

$$p_i = \frac{h(i)}{\sum_{j=0}^{L-1} h(j)} \tag{4.2}$$

où $h(i)$ est le nombre d'occurrences du pixel de niveau de gris $i \in \{1, \ldots, L-1\}$.

Dans le cas d'une binarisation, le seuil optimal t^* est celui qui maximise le ratio de la variance interclasse à la variance totale :

$$t^* = Arg \max_{th} \left(\frac{\sigma_B^{2(th)}}{\sigma_T^{2(th)}} \right) \tag{4.3}$$

avec :

$$\sigma_B^{2(th)} = P_{th} (P_{th} - 1) (\mu_1 - \mu_0)^2 \tag{4.4}$$

$$\sigma_T^{2(th)} = \sum_{i=1}^{L-1} p_i (i - \mu_T)^2 \tag{4.5}$$

où $\mu_0 = \sum_{i=0}^{th} i \frac{p_i}{P_t}$, $\mu_1 = \sum_{i=th+1}^{L-1} i \frac{p_i}{1-P_t}$, $\mu_T = \mu_0 + \mu_1$, $P_t = \sum_{i=0}^{th} p_i$ et $1 - P_t = \sum_{i=th+1}^{L-1} p_i$.

Comme la méthode d'Otsu consiste à regrouper les pixels en deux classes, l'efficacité de cette méthode n'est prouvée que lorsqu'il n'y a qu'un seul type d'objet à segmenter dans l'image. Une version multi-classes est disponible dans [Cheriet et al., 1998].

4.2.2.2.2 Méthode de Kapur et al. La méthode de Kapur et al., ou méthode du seuillage entropique, est basée sur le principe de la maximisation de l'entropie totale de Shannon. La méthode suppose que deux objets d'une même image ont des densités de probabilité indépendantes. Le seuil optimal est celui qui permet de maximiser l'entropie de l'image partitionnée. Dans le cas de deux classes, les entropies partielles et totales sont données par :

$$H_A(t) = -\sum_{i=0}^{t} \frac{p_i}{P_t} \log_2 \left(\frac{p_i}{P_t} \right) \tag{4.6}$$

$$H_B(t) = -\sum_{i=t+1}^{L-1} \frac{p_i}{1-P_t} \log_2 \left(\frac{p_i}{1-P_t} \right) \tag{4.7}$$

$$H_T(t) = H_A(t) + H_B(t) \tag{4.8}$$

Le problème de seuillage entropique est alors défini par :

$$t^* = Arg \max_{0 \leq t \leq L-1} (H_T(t)) \tag{4.9}$$

Cette méthode a ouvert le champ à la définition de nombreuses mesures d'information pour mieux segmenter les images, telles que l'entropie de Tsallis ou l'entropie de Renyi. Une description de ces variantes est disponible dans [Sezgin et al., 2004]. Une variante de cette méthode basée sur l'entropie exponentielle 2D est utilisée dans la section 4.3.

4.3 Segmentation d'images IRM basée sur l'entropie exponentielle 2D

4.3.1 Introduction

Avec le développement des nouvelles techniques d'imagerie, telles que l'imagerie IRM ou la tomodensitométrie, les médecins ont un besoin d'algorithmes de traitement d'images de plus en plus efficaces. Le but de l'imagerie médicale est d'apporter aux médecins des outils d'aide à la décision. Les images sont traitées afin d'extraire de celles-ci des données quantitatives, qui vont permettre aux médecins d'infirmer ou de confirmer leurs avis de spécialistes.

La segmentation occupe une place prépondérante dans le processus d'imagerie médicale, dans le sens où elle permet d'isoler certains objets sur les images, et donc de pouvoir mesurer les propriétés spatiales de ceux-ci. Dans cette section, nous présentons une méthode de segmentation de la matière blanche dans l'espace vésiculaire du cerveau. Les traitements sont réalisés sur des images IRM du cerveau. Des résultats sur ce genre d'images sont disponibles dans [Drapaca et al., 2005 ; Qiao et al., 2007 ; Warfield et al., 2000, Murgasova et al., 2006 ; Song et al., 2006 ; Kamber et al., 2000 ; Cocosco et al., 2003]. Le but est d'utiliser TRIBES pour optimiser un critère d'entropie exponentielle 2D, pour trouver les seuils optimaux de ces images dans un temps raisonnable.

4.3.2 Histogramme 2D

Soit f l'image originale et w l'image moyennée à partir d'un masque 3x3, tel que défini par l'équation (4.2) :

$$w(x,y) = \left\lfloor \frac{1}{9} \sum_{i=-1}^{1} \sum_{j=-1}^{1} f(x+i, y+j) \right\rfloor \tag{4.10}$$

L'histogramme 2D h d'une image [Nakib et al., 2007] est alors défini par :

$$h(i,j) = \frac{Card\,(f(x,y) = i \text{ et } w(x,y) = j)}{Taille\ de\ l'image} \tag{4.11}$$

avec $i, j \in \{0, 1, \ldots, L-1\}$, où L est le nombre de niveaux de gris disponibles. Dans la suite, on posera $L = 256$. On note la probabilité jointe $p_{ij} = h(i,j)$.

L'histogramme 2D est représenté par la Figure 4.1. Les cadrans 1 et 2 matérialisent respectivement le fond et les objets de l'image. Les cadrans 3 et 4 contiennent respectivement l'information concernant le bruit et les contours. Un vecteur de seuillage de l'histogramme 2D h est noté (s,t),

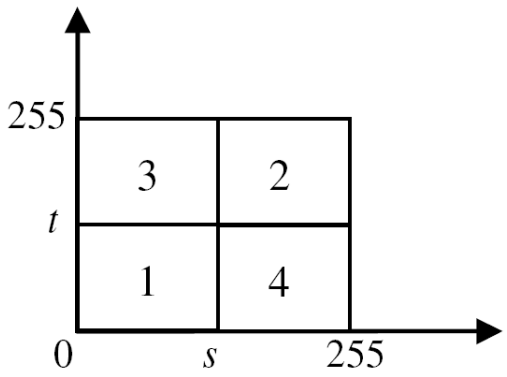


FIGURE 4.1 – Histogramme 2D.

où s représente le seuil de l'image moyennée w et t représente le seuil de l'image originale f. Dans le cas d'une segmentation de l'image en N classes, les classes sont données par :

$$P_{m-1}(a_{n-1}, a_n) = \sum_{i=s_{n-1}}^{s_n-1} \sum_{j=t_{n-1}}^{t_n-1} p_{ij} \tag{4.12}$$

$$P_m(a_n, a_{n+1}) = \sum_{i=s_n}^{s_{n+1}-1} \sum_{j=t_n}^{t_{n+1}-1} p_{ij} \tag{4.13}$$

où $a_n = (s_n, t_n)$, $n = 1, \ldots, N$; $m = 1, \ldots, N$, sont les vecteurs de seuillage.

4.3.3 Entropie exponentielle 2D

L'entropie exponentielle 2D (2DEE) est définie par :

$$H_\alpha = \left(\sum_i \sum_j p_{ij}^\alpha \right)^{\frac{1}{1-\alpha}} \tag{4.14}$$

avec $\alpha \in [0, 1[$.

L'entropie exponentielle 2D de la classe m d'une image segmentée est donnée par :

$$H_\alpha^{(m)}(a_n, a_{n+1}) = \left(\sum_{i=s_n}^{s_{n+1}-1} \sum_{j=t_n}^{t_{n+1}-1} \left(\frac{p_{ij}}{P_m(a_n, a_{n+1})} \right)^\alpha \right)^{\frac{1}{1-\alpha}} \tag{4.15}$$

Pour des raisons pratiques, on définit $T_0 = (s_0, t_0) = (0, 0)$ et $T_N = (s_N, t_N) = (255, 255)$. Le problème impose $t_0 < t_1 < \ldots < t_N$ et $s_0 < s_1 < \ldots < s_N$.

L'entropie 2D totale est donc donnée par :

$$H_\alpha^T(a_1, \ldots, a_{N-1}) = \sum_{i=0}^{N-1} H_\alpha^{(i+1)}(a_i, a_{i+1}) \tag{4.16}$$

En appliquant le principe du maximum de l'entropie, le vecteur de seuils optimaux $(a_1^*, \ldots, a_N^*) = \left((s_1^*, t_1^*), \ldots, (s_{N-1}^*, t_{N-1}^*)\right)$ doit satisfaire :

$$H_\alpha^T (a_1^*, \ldots, a_N^*) = \max \left(H_\alpha^T (a_1, \ldots, a_{N-1})\right) \tag{4.17}$$

avec $0 < t_1 < \ldots < t_{N-1} < 255$ et $0 < s_1 < \ldots < s_{N-1} < 255$.

Dans le cas d'une image à segmenter en deux classes (N=2), déterminer le vecteur optimal (s^*, t^*) demande une complexité en $\mathcal{O}\left(L^4\right)$, où L est le nombre de niveaux de gris. Dans le cas de N classes, la complexité est de $\mathcal{O}\left(L^{2N+2}\right)$. Dans la suite, nous proposons d'appliquer ce critère à la résolution du problème de maximisation d'entropie à l'aide de TRIBES.

Pour montrer l'intérêt de cette méthode, nous l'avons comparée à cinq autres méthodes classiques de calculs de seuils : un algorithme EM (EM) [Bazi et al., 2007], une méthode basée sur la méthode dite de "valley-emphasis" (VE) [Ng, 2006], la méthode d'Otsu [Sezgin et al., 2004], la méthode de Kapur et al. [Sezgin et al., 2004] et une méthode basée sur l'entropie 2D de Tsallis (TE) [Sahoo et al., 2006]. La comparaison est effectuée sur une image synthétique avec différents niveaux de bruit (cf. Figure 4.2). Les performances sont évaluées par le critère de mauvaise classification ME [Sezgin et al., 2004]. Ce critère est défini par une corrélation entre les images et la perception visuelle. Son expression analytique est donnée par (4.18) :

$$ME\,(\%) = \left(1 - \frac{|B_O \cap B_T| + |F_O \cap F_T|}{|B_0| + |F_0|}\right) \times 100 \tag{4.18}$$

où B_O et F_O sont respectivement le fond et l'objet de l'image originale, B_T et F_T sont respectivement le fond et l'objet de l'image seuillée. Dans le cas idéal, ME est égal à 0%.

La valeur du vecteur de seuils optimal dépend de l'ordre α de la 2DEE (*2D exponential entropy*). Afin de trouver l'ordre optimal α^*, on utilise le critère d'uniformité. Ce critère est défini par :

$$U\,(\alpha) = \left(\frac{1-2N}{M}\right) \sum_{j=0}^{N} \sum_{i \in C_j} \left(\frac{f_i - \mu_i}{f_{max} - f_{min}}\right)^2 \tag{4.19}$$

où :

- N est le nombre de seuils,
- C_j est la $j^{ème}$ classe ;
- M est le nombre de pixels de l'image ;
- f_i le niveau de gris du pixel i ;
- μ_j est le niveau de gris moyen de la classe j ;
- f_{max} et f_{min} sont respectivement le plus grand et le plus faible niveau de gris de l'image.

U est strictement positif et compris entre 0 et 1. Si U est proche de 1, il y a une forte uniformité dans l'image, et inversement.

Les résultats expérimentaux sont présentés dans la Table 4.1. TRIBES est utilisé pour optimiser le critère de 2DEE. On voit sur cette Table que le seuillage avec utilisation de l'entropie exponentielle donne de meilleurs résultats que les autres méthodes.

FIGURE 4.2 – (A) Image originale. (B) à (D) Images bruitées

TABLE 4.1 – Évaluation des différentes méthodes de seuillage.

	Méthodes de segmentation					
	Otsu *ME(%)*	Kapur *ME(%)*	EM *ME(%)*	VE *ME(%)*	TE *ME(%)*	2DEE *ME(%)*
Image B	0.45	5.61	8.68	0.34	0.64	**0.23** ($\alpha = 0.4$)
Image C	0.88	4.50	12.50	0.63	1.12	**0.56** ($\alpha = 0.4$)
Image D	12.22	4.97	28.87	11.59	12.90	**3.57** ($\alpha = 0.6$)

4.3.4 Résultats expérimentaux et discussion

La segmentation d'image est réalisée en maximisant l'entropie 2D de l'image à l'aide de TRIBES en fonction des vecteurs de seuils optimaux. Les expérimentations réalisées montrent que la meilleure position de l'essaim n'évolue plus beaucoup après 1000 évaluations de la fonction objectif (cf. Figure 4.3). Le critère d'arrêt de l'algorithme est donc $Max_{Fes} = 1000$.

Les résultats obtenus sont présentés sur la Figure 4.4. Cette figure montre les seuillages obtenus à l'aide de TRIBES pour un cas sain (a) et un cas pathologique (b) pour N=4 (c,d) et N=5 (e,f). Le but est le quantifier le taux de matière blanche située dans l'espace ventriculaire. La Figure 4.5 présente les mêmes images, mais en utilisant l'entropie de Shannon 2D classique (2DSE), optimisée à l'aide d'une OEP classique [Peng-Yeng, 2007].

Dans le cas d'une recherche exhaustive, le critère doit être évalué en $\left(\frac{L!}{(L+1-N)!(N-1)!}\right)^2$ points, où L est le nombre total de niveaux de gris de l'image. La Table 4.2 montre le gain en temps obtenu en utilisant TRIBES vis-à-vis d'une recherche exhaustive. De plus, en comparant avec d'autres méthodes de segmentation proposées dans la littérature [Drapaca et al., 2005 ; Qiao et al., 2007 ; Warfield et al., 2000, Murgasova et al., 2006 ; Song et al., 2006 ; Kamber et al., 2000 ; Cocosco et al., 2003], on constate que l'utilisation de TRIBES apporte aussi un gain en temps appréciable à la segmentation des images IRM, étant donné que ces méthodes nécessitent au moins 120 secondes pour donner un résultat.

Les Figures 4.4 et 4.5 montrent que la méthode utilisant la 2DEE présente un meilleur résultat de segmentation que la méthode utilisant une entropie classique. La matière blanche est beaucoup mieux segmentée sur les images de la Figure 4.4 que sur celles de la Figure 4.5.

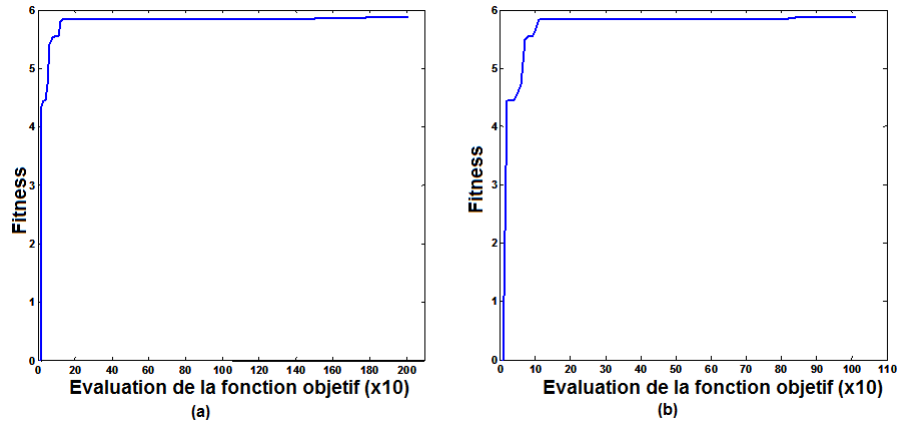


FIGURE 4.3 – Évolution de la valeur de l'entropie 2D en fonction du nombre d'évaluations de la fonction objectif. (a) $Max_{FEs} = 2000$, (b) $Max_{FEs} = 1000$.

TABLE 4.2 – Temps d'exécution pour segmenter l'image IRM de cerveau à l'aide de TRIBES.

Nombre de classes (N)	**Temps d'exécution (s)**	**Facteur de gain**
3	14.8	106e4
4	19.6	687e8
5	26.7	194e16

4.3.5 Conclusion

Dans cette section, nous avons montré comment TRIBES peut être utilisé pour résoudre un problème de segmentation d'images IRM. Le critère utilisé, l'entropie exponentielle, aboutit à des résultats de segmentation satisfaisants. L'utilisation de TRIBES permet un gain de temps important vis-à-vis d'une recherche exhaustive, mais aussi vis-à-vis d'autres méthodes de segmentation.

Ce travail a fait l'objet d'une publication [Nakib et al., 2007b].

4.4 Seuillage d'images par approximation d'histogramme

4.4.1 Introduction

Dans cette section, le processus de seuillage est abordé sous un aspect différent de celui vu dans la section précédente. Dans la section 4.3, le seuillage était réalisé directement à partir des informations obtenues à partir de l'histogramme, en calculant un critère. Dans la présente section, une méthode permettant d'approcher l'histogramme par une somme de fonctions gaussiennes est proposée. Un telle opération nous permet d'obtenir une expression analytique approchée de l'histogramme, et donc de pouvoir choisir les seuils optimaux plus facilement.

4.4.2 Approximation de l'histogramme

Soit h l'histogramme d'une image quelconque codée sur une échelle comportant L niveaux de gris. On cherche à simplifier le processus de seuillage en approchant h par une somme de d densités

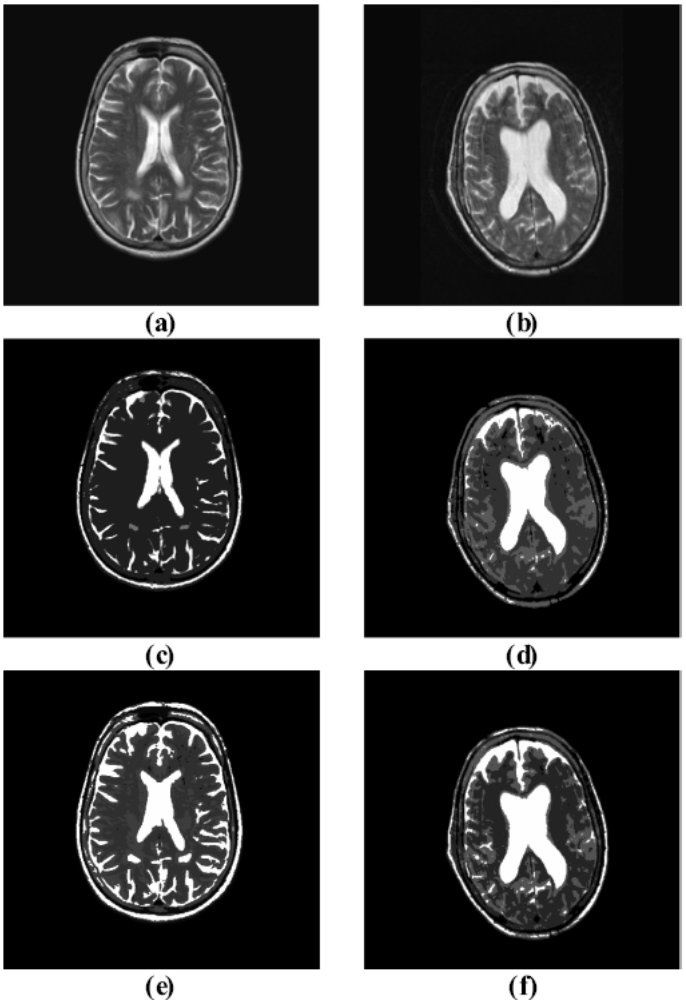


FIGURE 4.4 – Segmentation obtenue à l'aide de TRIBES suivant le critère de 2DEE. (a) Image originale cas sain, (b) Image originale cas pathologique, (c) Image saine segmentée avec N=4 et $\alpha = 0.1$, (d) Image pathologique segmentée avec N=4 et $\alpha = 0.3$, (e) Image saine segmentée avec N=5 et $\alpha = 0.4$, (f) Image pathologique segmentée avec N=5 et $\alpha = 0.2$.

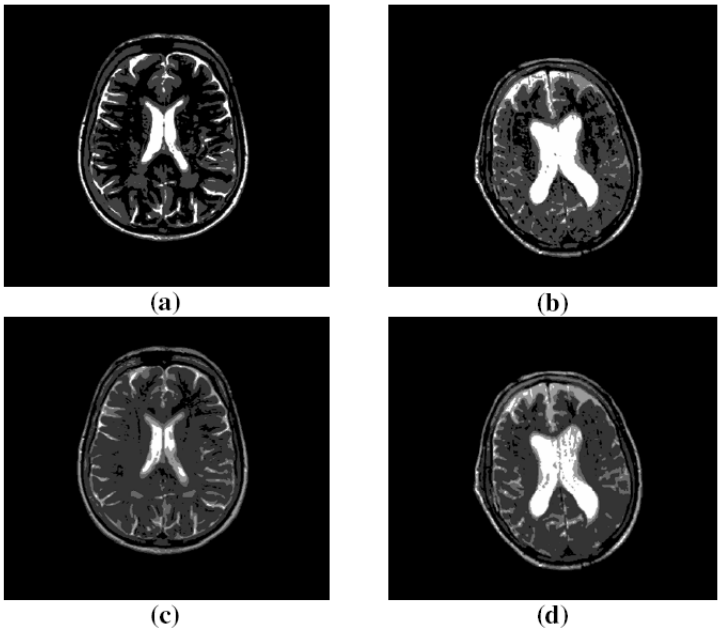


FIGURE 4.5 – Segmentation obtenue à l'aide d'une OEP classique et 2DSE. (a) Image saine segmentée avec N=4 , (b) Image pathologique segmentée avec N=4, (c) Image saine segmentée avec N=5, (d) Image pathologique segmentée avec N=5 .

de probabilité [Synder et al., 1990]. Dans notre cas, des densités de probabilité gaussiennes sont utilisées. L'approximation de h est donnée par :

$$h_{approx}(x) = \sum_{i=1}^{d} P_i e^{-\left(\frac{x-\mu_i}{\sigma_i}\right)^2} \tag{4.20}$$

où P_i est l'amplitude de la $i^{ème}$ gaussienne, μ_i est la moyenne et σ_i est la variance.

L'histogramme h est normalisé de la manière suivante :

$$h_{norm}(i) = \frac{h(i)}{\sum_{j=0}^{L-1} h(j)} \tag{4.21}$$

Approcher l'histogramme h avec la somme de gaussiennes h_{approx} revient donc à minimiser le critère J en fonction du jeu de paramètres Θ. L'expression du critère J est donnée par :

$$J = \frac{\sum_{i=0}^{L-1} |h_{norm}(i) - h_{approx}(\Theta, i)|}{\sum_{j=0}^{L-1} h_{norm}(j)} \tag{4.22}$$

Le jeu de paramètres Θ est donné par :

$$\Theta = \{(P_i, \mu_i, \sigma_i), i = 1, \ldots, d\} \tag{4.23}$$

Si l'on suppose que l'histogramme est correctement approché par la somme de gaussiennes, alors les seuils optimaux sont calculés en minimisant la probabilité de recouvrement des différentes gaussiennes. Pour deux gaussiennes successives, l'expression de cette probabilité est donnée par :

$$E(T_i) = P_i \int_{-\infty}^{T_i} p_i(x)\,dx + P_{i+1} \int_{T_i}^{+\infty} p_{i+1}(x)\,dx,\ i = 1, \ldots, d-1 \tag{4.24}$$

où T_i est le $i^{ème}$ seuil de l'image et p_i est la $i^{ème}$ gaussienne de l'approximation.

Minimiser cette erreur revient à différencier $E(T_i)$ selon le critère de Leibniz en fonction de T_i et de l'égaler à zéro. Cela donne :

$$P_i p_i(x) = P_{i+1} p_{i+1}(x) \tag{4.25}$$

Dans le cas de densités de probabilités gaussiennes, on aboutit à la résolution d'une équation du second ordre, dont l'expression est donnée par :

$$AT_i^2 + BT_i + C = 0 \tag{4.26}$$

avec :

$$A = \sigma_i^2 + \sigma_{i+1}^2 \tag{4.27}$$

$$B = 2\left(\mu_i \sigma_{i+1}^2 - \mu_{i+1}\sigma_i^2\right) \tag{4.28}$$

$$C = \mu_{i+1}^2 \sigma_i^2 - \mu_i^2 \sigma_{i+1}^2 + 4\sigma_i^2 \sigma_{i+1}^2 \ln\left(\frac{\sigma_{i+1} P_i}{\sigma_i P_{i+1}}\right) \tag{4.29}$$

Cette expression a deux solutions possibles mais seulement une seule solution est acceptable dans le cas présent [Gonzalez et al., 2002].

4.4.3 Résultats expérimentaux de la méthode de segmentation par approximation de l'histogramme

La méthode est testée sur les images *Lena* (Figure 4.6 (a)) et *Boulons* (Figure 4.7 (a)). Ces images sont de taille 256x256, codées sur 256 niveaux de gris. Le critère J est optimisé à l'aide de TRIBES et de *Standard PSO 2006*. Le critère d'arrêt utilisé est $Max_{Fes} = 10000 \cdot d$. Les résultats obtenus avec TRIBES pour d=3, 4 et 5 sont respectivement présentés dans les Tables 4.3 et 4.4. Les résultats obtenus avec le *Standard PSO 2006* pour d=3, 4 et 5 sont respectivement présentés dans les Tables 4.5 et 4.6. On observe que, dans les cas d=3 et d=4, TRIBES et *Standard PSO 2006* donnent des résultats similaires. Dans le cas d=5, les résultats sont légèrement différents. Le problème augmentant en taille, il est plus difficile pour les deux algorithmes de trouver une solution correcte, donc de trouver les mêmes résultats.

La Figure 4.6 présente les résultats obtenus à l'aide de TRIBES dans le cas de l'image Lena. Les Figures 4.6 (b) et (c) montrent l'approximation de l'histogramme obtenue avec les paramètres optimaux fournis par TRIBES, dans les cas d=3 et d=4. Les Figures 4.6 (d) et (e) présentent les résultats de segmentation. On voit sur ces images que la segmentation de Lena en trois classes est insuffisante, mais semble convenable avec 4 classes.

La Figure 4.7 présente les résultats obtenus à l'aide de TRIBES dans le cas de l'image Boulons. Les Figures 4.7 (b) et (c) montrent l'approximation de l'histogramme obtenue avec les paramètres optimaux fournis par TRIBES, dans les cas d=3 et d=4. Les Figures 4.7 (d) et (e) présentent les résultats de segmentation.

Les Figures (e) et (f) des Figures 4.6 et 4.7 montrent les graphes de convergence de TRIBES. L'allure de ces graphes nous permet de confirmer la pertinence de notre critère d'arrêt.

TABLE 4.3 – Résultats expérimentaux sur l'image *Lena* obtenus avec TRIBES.

Nombre de gaussiennes	Paramètres des gaussiennes	Seuils	Erreur finale
3	P :(0.2624, 0.7685, 0.8153) μ :(198, 23, 100) σ :(14.0789, 8.3667, 66,4781)	T :(41, 179)	2.0761
4	P :(0.3541, 0.2902, 0.7758, 0.7547) μ :(133,196, 94, 23) σ :(8.9784, 16.8357, 64.3285, 8.6236)	T :(41, 119, 174)	1.2141
5	P :(0.3370, 0.2877, 0.7868, 0.3420,0.7536) μ :(65, 196, 23, 133, 98) σ :(4.7779, 16.3593, 9.3536, 8.5447, 63.0961)	T :(42, 75, 120, 175)	0.7023

TABLE 4.4 – Résultats expérimentaux sur l'image *Boulons* obtenus avec TRIBES.

Nombre de gaussiennes	Paramètres des gaussiennes	Seuils	Erreur finale
3	P :(0.570, 0.036, 0.529) μ :(34, 98, 251) σ :(7.383, 100.0, 0.614)	T :(55, 250)	0.3701
4	P :(0.035, 0.528, 0.845, 0.505) μ :(100, 251, 32,35) σ :(100.0, 0.616, 0.272, 7.866)	T :(34, 57, 250)	0.1311
5	P :(0.038, 0.537, 0.018, 0.639, 0.506) μ :(79, 251, 216, 32, 34, 57) σ :(56.781, 0.552, 100.0, 0.189, 7.891)	T :(34, 56, 170, 250)	0.1199

TABLE 4.5 – Résultats expérimentaux sur l'image *Lena* obtenus avec *Standard PSO 2006*.

Nombre de gaussiennes	Paramètres des gaussiennes	Seuils	Erreur finale
3	P :(0.262, 0.764, 0.815) μ :(198, 23, 100) σ :(14.078, 8.835, 66.480)	T :(41, 179)	2.0761
4	P :(0.775, 0.754, 0.290, 0.354) μ :(94, 23, 196, 133) σ :(64.328, 8.623, 16.835, 8.978)	T :(41, 119, 172)	1.2141
5	P :(0.337, 0.342, 0.786, 0.287, 0.753) μ :(65, 133, 23, 196, 98) σ :(4.777, 8.544, 9.353, 16.359, 63.095)	T :(42, 75, 119, 174)	0.7023

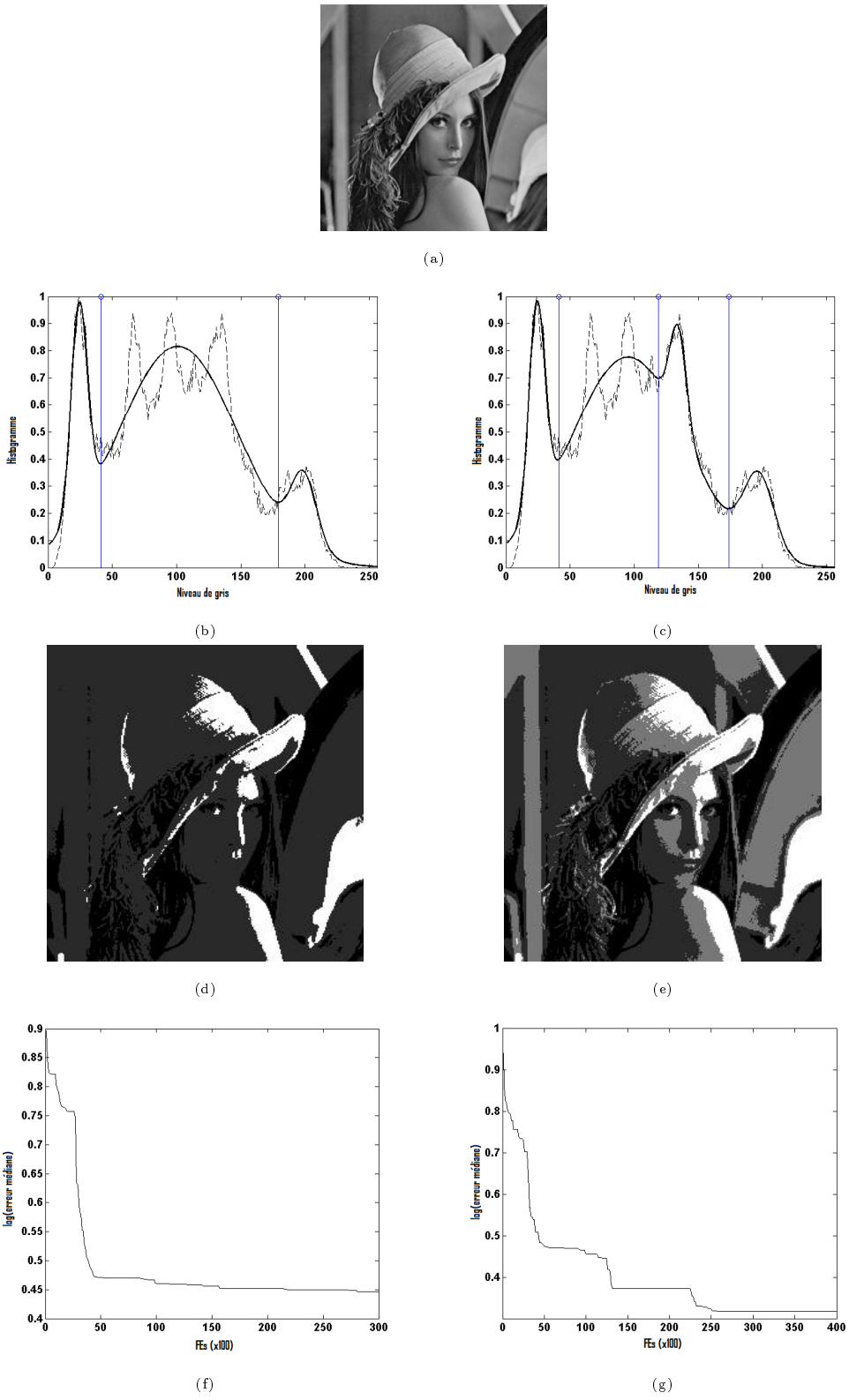
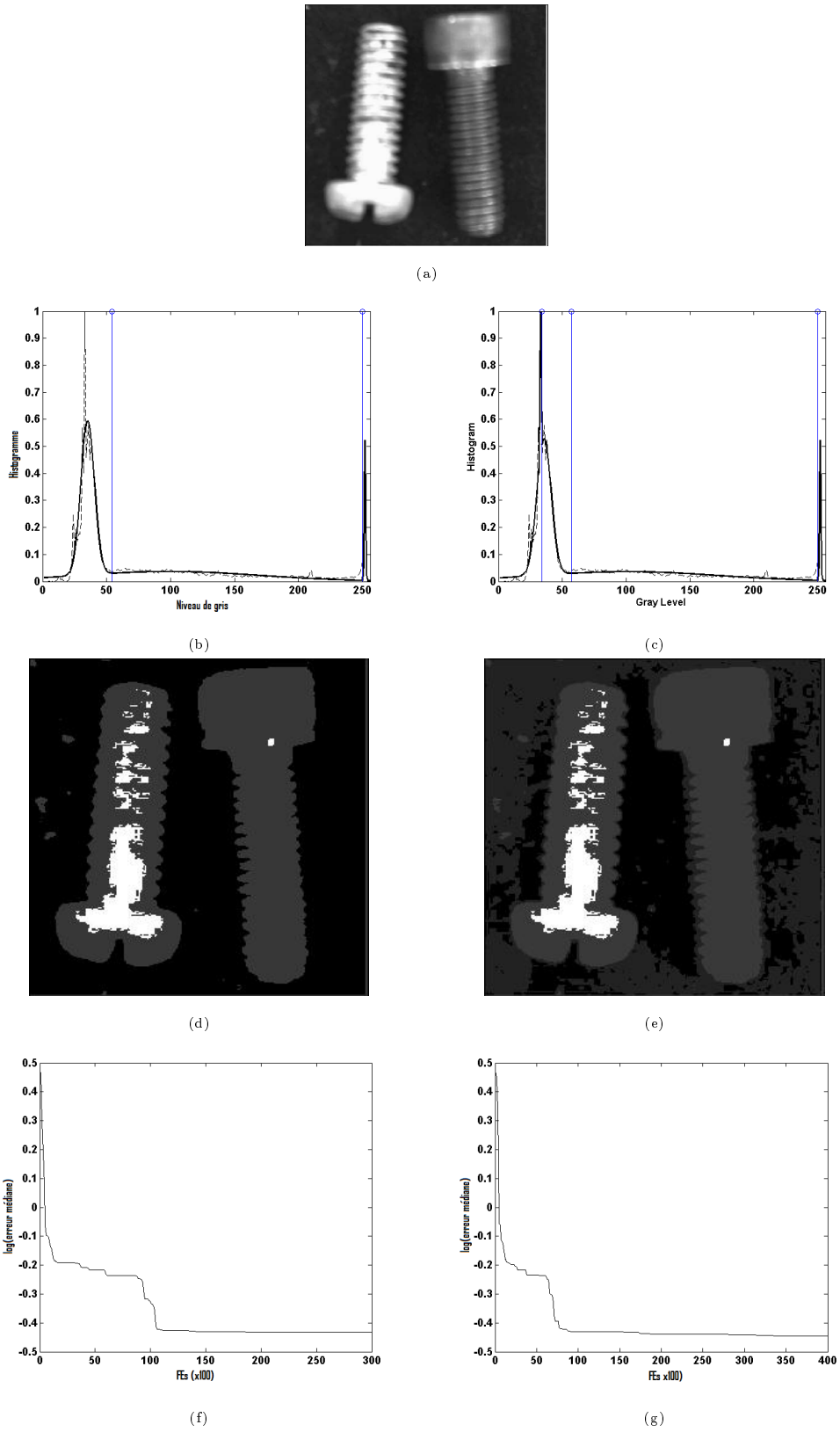


FIGURE 4.6 – Résultat de segmentation de l'image *Lena*. (a) Image originale, (b) Histogrammes original et approché pour d=3, (c) Histogrammes original et approché pour d=4, (d) Image segmentée pour d=3, (e) Image segmentée pour d=4, (f) Graphe de convergence pour d=3, (g) Graphe de convergence pour d=4.



(a) (b) (c) (d) (e) (f) (g)

FIGURE 4.7 – Résultat de segmentation de l'image *Boulons*. (a) Image originale, (b) Histogrammes original et approché pour d=3, (c) Histogrammes original et approché pour d=4, (d) Image segmentée pour d=3, (e) Image segmentée pour d=4, (f) Graphe de convergence pour d=3, (g) Graphe de convergence pour d=4.

TABLE 4.6 – Résultats expérimentaux sur l'image *Boulons* obtenus avec *Standard PSO 2006*.

Nombre de gaussiennes	Paramètres des gaussiennes	Seuils	Erreur finale
3	P :(0.5707, 0.0366, 0.5294) μ :(34, 98, 251) σ :(7.38, 100.0, 0.6141)	T :(54, 250)	0.3701
4	P :(0.5293, 0.0358, 0.5252, 0.5058) μ :(251, 101, 32, 35) σ :(0.6135, 100.0, 0.1752, 7.8581)	T :(34, 57, 250)	0.1311
5	P :(0.5284, 0.0182, 0.5070, 0.0384, 0.7295) μ :(251, 214, 34, 79, 32,) σ :(0.5644, 100.0, 7.8907, 56.5071, 0.2354)	T :(34, 56, 170, 250)	0.1199

4.4.4 Conclusion

Dans cette section, nous avons présenté une méthode de segmentation basée sur l'approximation de l'histogramme original par une somme de densités de probabilité. Obtenir, par approximation, une expression analytique de l'histogramme permet ainsi de calculer beaucoup plus facilement les seuils optimaux. Les résultats proposés pour les images *Lena* et *Boulons* montrent que cette méthode, couplée à l'utilisation de TRIBES, offre une alternative efficace et rapide aux autres méthodes de segmentation.

Ce travail a fait l'objet d'une publication [Cooren et al., 2008c].

4.5 Conclusion

Le traitement d'images est aujourd'hui un enjeu important dans le monde de la recherche. Considérant que ses domaines d'applications sont nombreux et variés, c'est un sujet qui est en pleine ébullition et qui profite totalement des avancées technologiques en matière d'acquisition des images et de puissance des calculateurs. Parmi les nombreux axes de recherche au sein de la communauté de l'imagerie, la segmentation fait partie des sujets les plus actifs. Segmenter une image consiste à isoler les différents objets ou les différentes zones de cette image. Segmenter les différentes parties d'une image permet de pouvoir calculer les propriétés de ces zones d'intérêt à partir de l'information apportée par l'intensité lumineuse.

La méthode de seuillage est une technique particulière de segmentation. Elle repose sur l'hypothèse que les différents objets de l'image sont différenciables uniquement par leurs intensités lumineuses. En pratique, cela implique que la segmentation de l'image va se déterminer à partir de l'histogramme de l'image. Aucune notion de voisinage n'est utilisée ici.

L'entropie exponentielle est un critère qui permet de trouver les seuils optimaux d'une image à partir de son histogramme. Les seuils optimaux sont trouvés quand l'entropie exponentielle est maximale. Le problème de seuillage devient alors un problème d'optimisation qui peut être résolu par n'importe quelle métaheuristique.

Les résultats présentés montrent que TRIBES parvient parfaitement à résoudre ce problème dans le cas d'images IRM de cerveau. De plus, la segmentation est réalisée dans un temps beaucoup plus rapide qu'avec les méthodes classiques.

Ce chapitre présente enfin comment l'histogramme d'une image peut être approché par une expression analytique. L'intérêt de disposer d'une telle expression est de pouvoir déterminer théori-

quement les seuils optimaux. L'utilisation de TRIBES conduit ici aussi à des résultats intéressants, qui offrent une alternative crédible aux méthodes d'imagerie pure.

Chapitre 5

Application de TRIBES et de MO-TRIBES au dimensionnement de circuits électroniques

5.1 Introduction

Les avancées en matière de circuits intégrés VLSI ("Very Large Scale Integration") permettent la réalisation de circuits intégrés de haute complexité [Gielen et al., 1991]. Les composants analogiques tiennent une part importante dans la conception de tels circuits, tant pour les systèmes à signal mixte que pour les systèmes numériques [Gielen et al., 1991 ; Toumazou et al., 1993]. La conception des composants analogiques apparaît donc comme une étape importante lors de la réalisation de tels circuits. En effet, les performances de ces circuits sont directement dépendantes des paramètres des composants analogiques. La conception de circuits analogiques est un processus complexe, dans le sens où elle ne concerne pas seulement le placement et le routage des composants, mais aussi leur dimensionnement [Graeb et al., 2001]. Dans la majorité des cas, le dimensionnement des composants analogiques est un processus lent, fastidieux et itératif, qui est réalisé grâce à l'expérience et l'intuition d'un expert [Conn et al., 1996]. L'enjeu est d'essayer de proposer des méthodes automatiques de dimensionnement des composants analogiques, afin d'accélérer de manière efficace le processus de conception des circuits intégrés. Les approches les plus connues dans la littérature sont basées sur des topologies fixes et/ou des techniques stochastiques [Medeiro et al., 1994 ; Silveira et al., 1996 ; Conn et al., 1996 ; Loulou et al., 2002]. Ces méthodes sont, dans la majorité des cas, initialisées avec une "bonne" solution fournie par un expert en conception de circuits analogiques. Ces méthodes cherchent ensuite à améliorer cette solution par différentes techniques. Le problème de ces méthodes est qu'elles sont souvent très lentes et qu'elles ne garantissent pas la convergence vers un optimum global.

Si l'on aborde le problème de dimensionnement du point de vue non pas de l'électronique, mais plutôt de la recherche opérationnelle, on s'aperçoit que le problème de dimensionnement est un problème d'optimisation à variables continues soumis à des contraintes. Les variables de décision sont alors la largeur et la longueur des transistors du circuit, les contraintes étant imposées par le cahier des charges du circuit.

Dans ce chapitre, nous présentons deux applications de TRIBES au dimensionnement de circuits analogiques, l'une mono-objectif et l'autre multiobjectif. Dans un premier temps, nous décrivons comment TRIBES peut être utilisé pour dimensionner les transistors d'un amplificateur de tension faible bruit, afin de maximiser le gain de celui-ci. Dans un deuxième temps, nous montrons comment

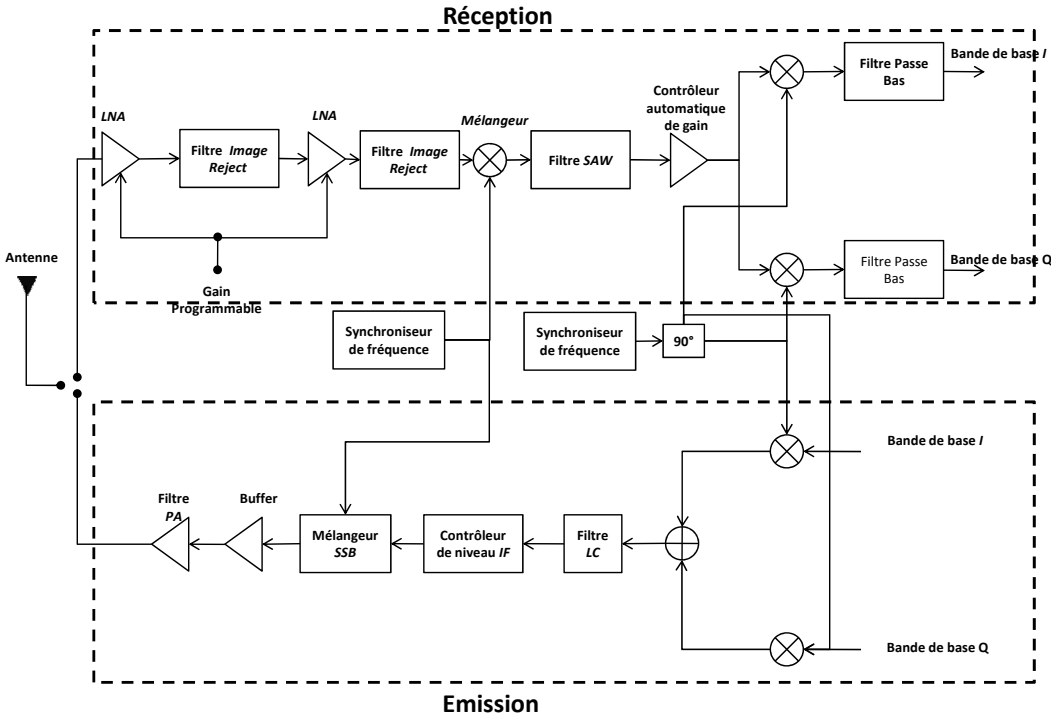


FIGURE 5.1 – Chaîne de transmission radio-fréquence.

optimiser les performances d'un convoyeur de courant de deuxième génération à l'aide de MO-TRIBES.

Cette application ayant été réalisée en cours de thèse, elle permet de corroborer les observations faites sur le comportement de TRIBES et, justifie donc les recherches d'améliorations pour TRIBES initiées avec TRIBES+. De même, cette application nous permet d'affiner nos conclusions sur le comportement global de MO-TRIBES.

5.2 Amplificateur faible bruit

Un amplificateur faible bruit (LNA, *Low Noise Amplifier*) est un dispositif électronique utilisé dans les systèmes de communication. Il sert à amplifier les signaux très faibles captés par les antennes tout en réduisant le bruit. Dans la majorité des cas, un tel amplificateur est placé à proximité de l'antenne, afin d'éviter les pertes en ligne. Pour cela, il est aussi souvent nommé *préamplificateur*. La Figure 5.1 situe la place d'un amplificateur faible bruit dans la chaîne de transmission radio-fréquence. La Figure 5.2 présente l'amplificateur faible bruit utilisé dans cette section.

On est ici en présence d'un amplificateur à source commune avec dégénérescence inductive (L_s). Le transistor M_1 améliore la stabilité du circuit en minimisant la contre-réaction entre la sortie et l'entrée [Razavi, 1998]. L'impédance d'entrée est principalement due aux inductances L_g et L_s, mais aussi à la capacité grille-source C_{gs} du transistor M_1. L'impédance de sortie correspond au circuit LC (L_{ch}, C_{ch}) et à l'impédance équivalente des transistors M_1 et M_2. Le sous-circuit composé par les éléments M_3, R_1 et R_2 est destiné à polariser le circuit. Le courant de polarisation I_d est ici égal à 29 mA.

Pour assurer une transmission maximale de la puissance du signal de l'antenne à l'amplificateur

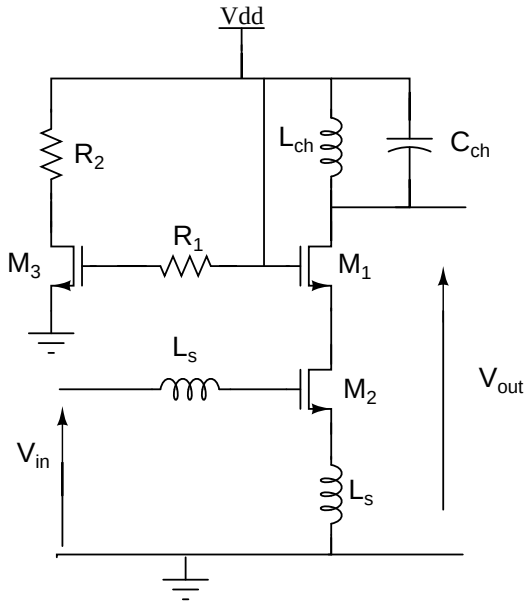


FIGURE 5.2 – Amplificateur faible bruit.

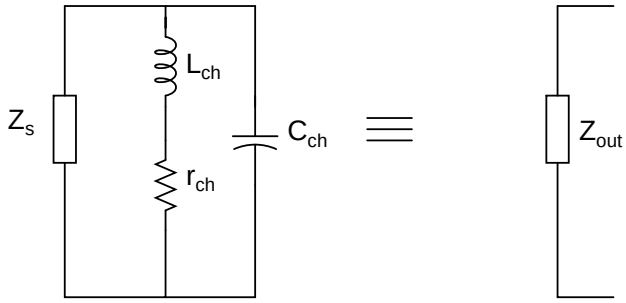


FIGURE 5.3 – Impédance de sortie de l'amplificateur faible bruit.

faible bruit, l'impédance d'entrée de l'amplificateur doit être égale à celle de sortie de l'antenne, qui en général est égale à 50 Ω. L'impédance d'entrée de l'amplificateur faible bruit est donnée par :

$$Z_{in}(j\omega) = \frac{1}{j\omega C_{gs}} + j\omega (L_g + L_s) + \frac{g_m L_s}{C_{gs}} \tag{5.1}$$

où g_m est la transconductance du transistor M_1 et C_{gs} est la capacité grille-source du transistor M_1.

De même, l'impédance de sortie doit être égale à l'impédance d'entrée du bloc suivant. Cette impédance est donnée par le schéma de la Figure 5.3.

r_{ch} est la résistance parasite de l'inductance L_{ch}.

Z_s est l'impédance équivalente des transistors M_1 et M_2.

Son expression est donnée par :

$$Z_{out} = r_{ds1} r_{ds2} g_{m2} \tag{5.2}$$

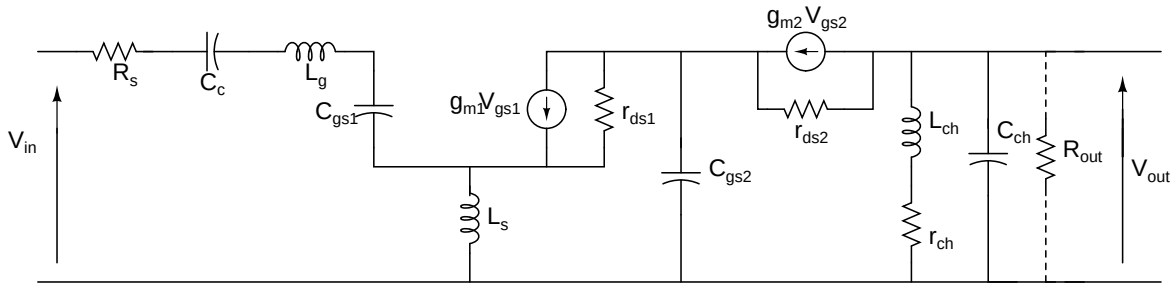


FIGURE 5.4 – Modèle petit-signal de l'amplificateur faible bruit.

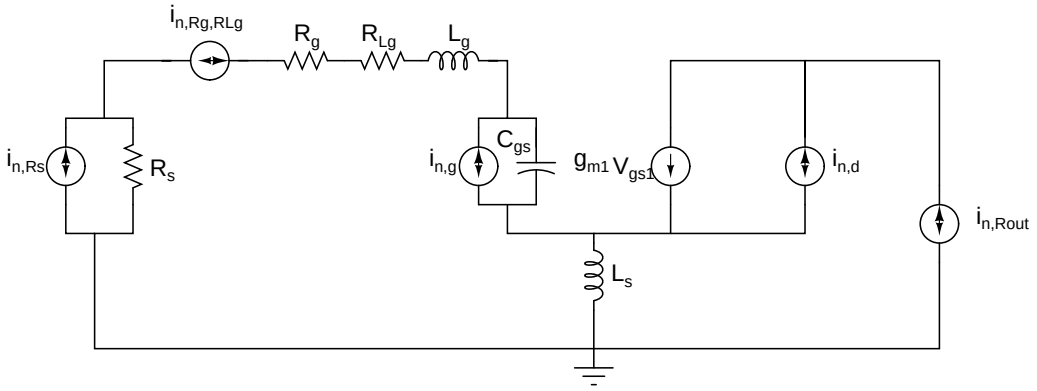


FIGURE 5.5 – Sources de bruit dans un amplificateur faible bruit.

où r_{ds1} et r_{ds2} sont respectivement les résistances drain-source des transistors M_1 et M_2, et g_{m2} est la transconductance du transistor M_2.

Le modèle petit-signal de l'amplificateur est donné par la Figure 5.4.

L'étude du bruit imposé par ce circuit est réalisée suivant le modèle établi dans [Andreani et al., 2001]. Le schéma équivalent utilisé pour calculer le facteur de bruit est présenté Figure 5.5.

L'expression du facteur de bruit est donnée par [Hung et al., 1990 ; Andreani et al., 2001] :

$$F = 1 + \frac{\beta_1 \left(Q^2 + \frac{1}{4}\right) P^2 \frac{g_m^2}{g_{dn}} + \frac{\gamma_1}{4} g_{dn} + \sqrt{\frac{\gamma_1 \beta_1}{4}} cP g_m + (R_s + R_L) Q^2 g_m^2 + \frac{1}{R_{out}}}{R_s Q^2 g_m^2} \tag{5.3}$$

avec $c = 0.4$ [Hung et al., 1990], $P = \frac{C_{gs}}{C_t}$, $\beta = \frac{4}{15}$ $g_{dn} = \frac{2}{3}$ g_{d0} et $Q = \frac{1}{\left(R_s + \frac{g_m L_s}{C_{gs}}\right)\omega_0 C_{gs}}$.

Assurer le comportement linéaire du circuit impose de satisfaire les inégalités suivantes :

$$V_{DS1min} > V_{Dsat1} \tag{5.4}$$

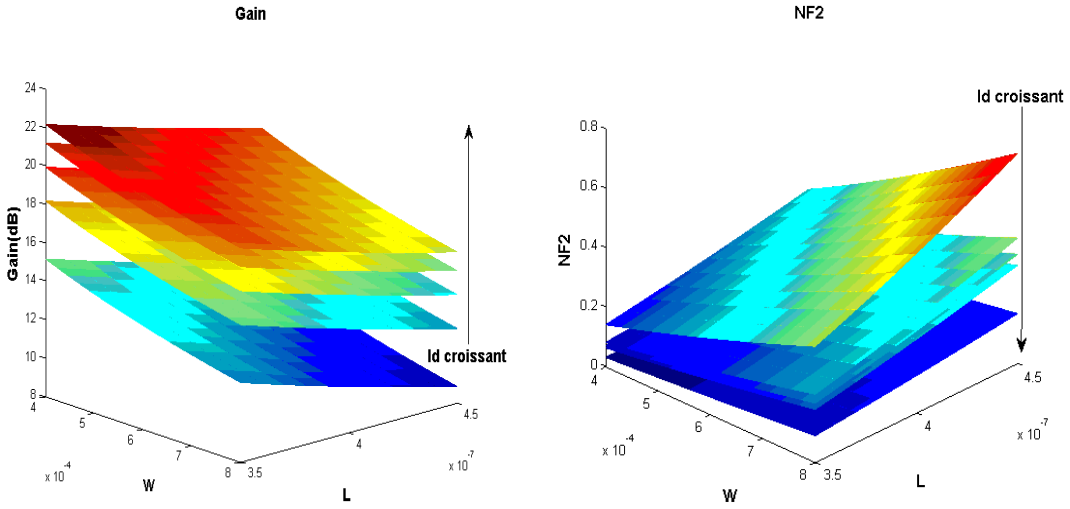
$$V_{DS2min} > V_{Dsat2} \tag{5.5}$$

où V_{DSimin} et V_{Dsati} sont la tension drain-source minimale et la tension de saturation du transistor M_i.

Les limites technologiques imposent :

TABLE 5.1 – Valeurs optimales des paramètres.

C_c(pF)	L_g(nH)	L_s(nH)	L_{ch} (nH)
10	8.110	0.320	0.715
r_{ch} (Ω)	C_{ch} (pF)	R_1(kΩ)	R_2 (kΩ)
1.92	7.44	1	1.03
W_1/L_1 (μm)	W_2/L_2 (μm)	W_3/L_3 (μm)	
510/0.35	500/0.35	40/0.35	



(a) (b)

FIGURE 5.6 – Variation du gain (a) et de la figure de bruit (b) en fonction du courant de polarisation.

$$35.10^{-8} \leq L \leq 45.10^{-8} \tag{5.6}$$

$$400.10^{-6} \leq W \leq 800.10^{-6} \tag{5.7}$$

où L et W sont respectivement la longueur et la largeur des transistors.

Enfin, on impose que la fréquence de transition f_T soit cinq fois supérieure à la fréquence UMTS $F_0 = 2.14\ GHz$.

Le but de cette application est de dimensionner les transistors M_1, M_2 et M_3 de l'amplificateur faible bruit afin que celui-ci présente un gain maximal à la fréquence UMTS F_0. Le gain est calculé en utilisant une technique de calcul symbolique à partir du schéma petit signal de la Figure 5.4. Son expression présente un très grand nombre de termes et il n'a pas été jugé utile de la présenter ici. Le problème est résolu en utilisant TRIBES. Le critère d'arrêt utilisé est $Max_{Fes} = 10000$. Les variations du gain en dB et de la figure de bruit [Andreani et al., 1991] en fonction du courant de polarisation I_d sont présentées Figure 5.6. La figure de bruit mesure la dégradation du rapport signal à bruit introduit par le circuit dans la chaîne de transmission radiofréquence. Les paramètres optimaux trouvés sont présentés dans la Table 5.1 pour $I_d = 29mA$.

Les simulations sont réalisées à l'aide du logiciel ADS, en utilisant la technologie ADS 0.35 μm, une alimentation de 2.5 V et un courant de polarisation I_d égal à 29 mA. Les Figures 5.7 et 5.8

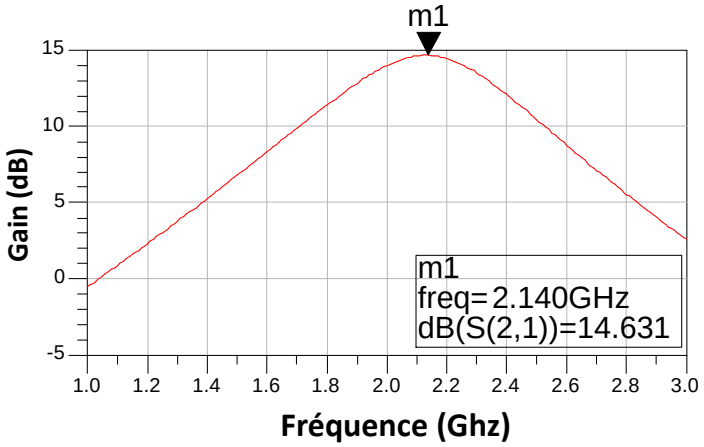


FIGURE 5.7 – Gain en tension en fonction de la fréquence.

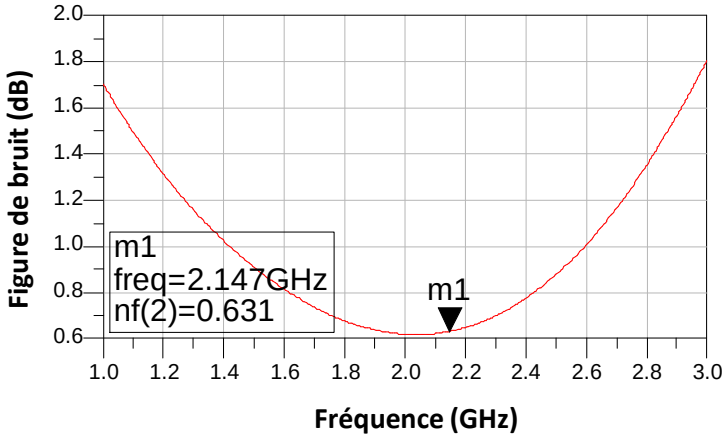


FIGURE 5.8 – Figure de bruit en fonction de la fréquence.

montrent les résultats obtenus par simulation avec les paramètres de la Table 5.1. Pour la fréquence UMTS F_0, on obtient un gain maximum égal à 14.631 dB et une figure de bruit quasi-minimale égale à 0.631 dB.

Cette étude a fait l'objet d'une publication lors de la conférence *European Conference on Circuits and Systems for Communications 2008* [Boughariou et al., 2008].

5.3 Convoyeur de courant de seconde génération

Un convoyeur de courant est un dispositif électronique composé des trois ports actifs. La représentation conventionnelle est donnée sur la Figure 5.9 (a). La Figure 5.9 (b) donne le schéma équivalent nullateur/norateur qui reproduit le comportement idéal d'un convoyeur de courant [Schmid, 2000]. La Figure 5.9 (c) donne le schéma réel, en prenant en compte les éléments parasites.

Les différents courants et tensions aux trois bornes X, Y et Z sont liés par la relation suivante [Toumazou et al., 1993 ; Ben Salem et al., 2006] :

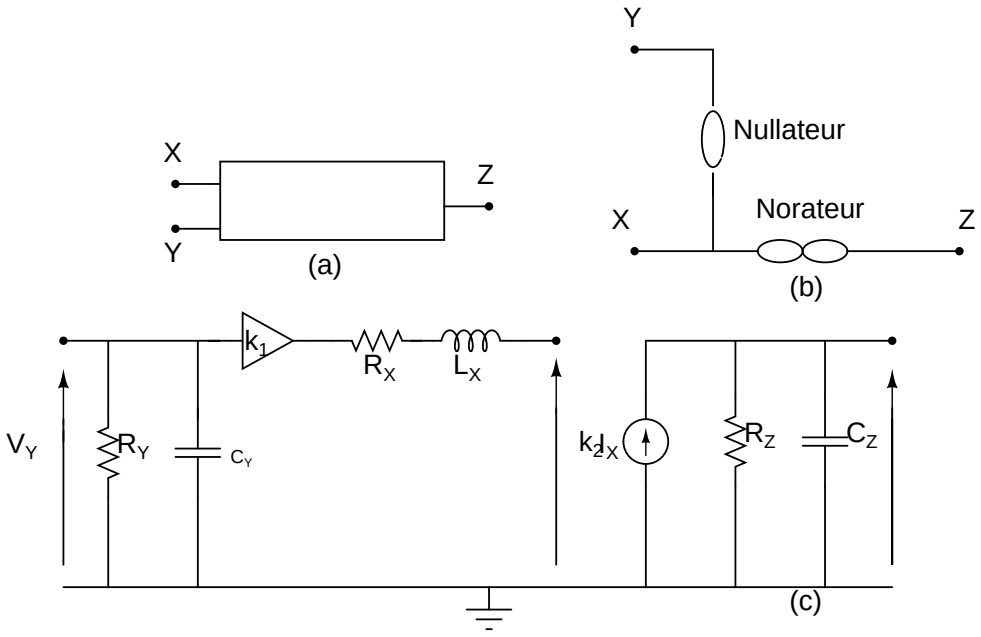


FIGURE 5.9 – (a) Représentation générale d'un convoyeur de courant, (b) Schéma idéal, (c) Schéma équivalent avec composants parasites.

$$\begin{pmatrix} I_Y \\ V_X \\ I_Z \end{pmatrix} = \begin{pmatrix} 0 & k_2 & 0 \\ 1 & 0 & 0 \\ 0 & k_1 & 0 \end{pmatrix} \begin{pmatrix} V_Y \\ I_X \\ V_Z \end{pmatrix} \tag{5.8}$$

Dans l'équation ci-dessus, k_2 spécifie le type de convoyeur de courant utilisé. Si $k_2 = 1$, alors le circuit est considéré comme un convoyeur de courant de première génération (CCI). A l'opposé, si $k_2 = 0$, le circuit est considéré comme un convoyeur de courant de seconde génération (CCII). k_1 caractérise le transfert de courant entre les ports X et Z. Si $k_1 = 1$, le circuit est considéré comme un convoyeur de courant *positif*. Le circuit est classé comme un convoyeur de courant *négatif* si $k_1 = -1$.

Un convoyeur de courant de seconde génération assure deux fonctionnalités :

- suiveur de courant entre les ports X et Z ;
- suiveur de tension entre les ports X et Y.

Dans le cas idéal, et pour assurer les bonnes fonctionnalités du circuit, un convoyeur de courant de deuxième génération est caractérisé par une faible impédance sur le port X et de hautes impédances sur les ports Y et Z.

Le convoyeur de courant de seconde génération est le convoyeur le plus souvent utilisé tant pour ses bonnes performances que pour l'intérêt apporté par l'utilisation d'une boucle translinéaire [Seevinck, 2000]. La Figure 5.10 montre l'exemple d'un convoyeur de courant de seconde génération *positif* à boucle translinéaire CMOS [Ben Salem et al., 2006]. Les transistors M_1 à M_4 constituent la boucle translinéaire qui assure $V_X = V_Y$. I_0 est le courant de polarisation et les transistors M_9 à M_{13} constituent des miroirs de courant. Les transistors M_5 à M_8 servent à reproduire au port Z le courant appliqué au port X.

Lorsqu'il est conçu au niveau microélectronique, le comportement d'un convoyeur de courant peut

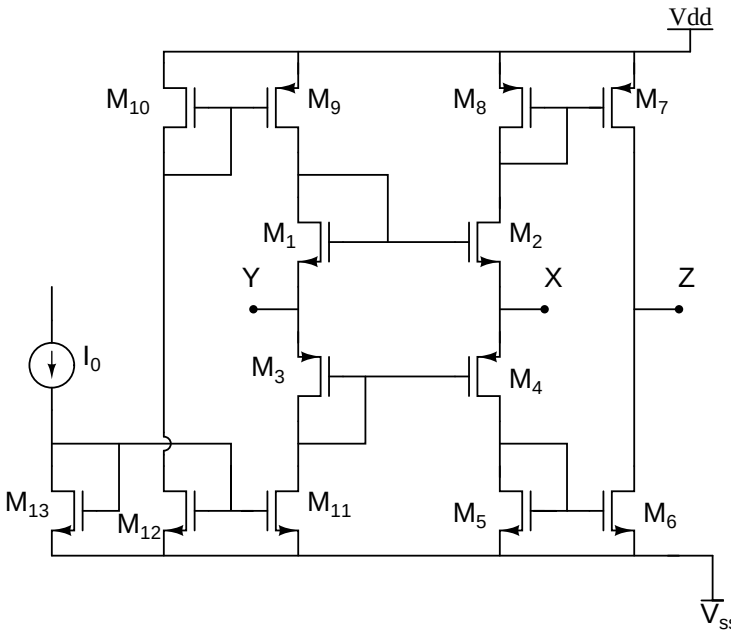


FIGURE 5.10 – Convoyeur de courant de seconde génération *positif*.

différer du cas idéal. La Figure 5.9 (c) montre les différents composants parasites d'un convoyeur de courant. Ces composants affectent de manière significative le comportement du circuit. Il a été prouvé que, parmi ces éléments, R_X est le composant parasite dont les effets sont les plus importants sur le comportement du convoyeur de courant.

Dans le cas d'applications hautes fréquences, ces circuits doivent être précisément dimensionnés, si l'on veut que les performances soient en accord avec les souhaits du concepteur. La bande passante en courant limite le domaine d'application du circuit, car la bande passante en courant est moins large que celle en tension. Donc, dans cette section, nous cherchons à optimiser les caractéristiques importantes du circuit, i.e. la résistance parasite R_X du port X et la fréquence de coupure en courant f_{ci}.

On peut montrer que l'expression de la résistance parasite R_X est donnée par [Ben Salem et al., 2006] :

$$R_X = \frac{1}{\sqrt{2K_N\left(\frac{W_N}{L_N}\right)(1+\lambda_N V_{DS})\,I_0} + \sqrt{2K_P\left(\frac{W_P}{L_P}\right)(1+\lambda_P V_{DS})\,I_0}} \tag{5.9}$$

où λ_N, λ_P, $K_N = 0.9386.10^{-8}$ et $K_P = 0.3476.10^{-6}$ sont des paramètres liés à la technologie des transistors (AMS 0.35 μm). I_0 est le courant de polarisation et V_{DS} est la tension drain-source des transistors. Dans notre cas, on utilise $I_0 = 100\ \mu A$. W_N et L_N sont respectivement la largeur et la longueur des transistors NMOS. W_P et L_P sont respectivement la largeur et le longueur des transistors PMOS.

L'expression de la fréquence de coupure en courant f_{ci} est calculée à partir d'une approche symbolique. Son expression n'est pas donnée ici du fait de son très grand nombre de termes.

Tous les transistors doivent opérer en mode saturation. Les contraintes de saturation sont données par :

TABLE 5.2 – Valeurs optimales des paramètres du circuit.

L_N (μm)	W_N (μm)	L_P (μm)	W_P (μm)	f_{cli} (GHz)	R_X (Ω)
0.55	7.29	0.35	12.6	1802	633
0.55	14.73	0.36	24.15	1253	448
0.57	10.63	0.35	17.06	1478	525
0.57	12.93	0.35	20.69	1330	477
0.59	8.27	0.35	13.87	1610	604
0.55	7.76	0.35	12.25	1771	610

$$\frac{V_{DD}}{2} - V_{TN} - \sqrt{\frac{I_0}{K_N \frac{W_N}{L_N}}} > \sqrt{\frac{I_0}{K_P \frac{W_P}{L_P}}} \tag{5.10}$$

$$\frac{V_{DD}}{2} - V_{TP} - \sqrt{\frac{I_0}{K_P \frac{W_P}{L_P}}} > \sqrt{\frac{I_0}{K_N \frac{W_N}{L_N}}} \tag{5.11}$$

où $V_{TN} = 0.4655\ V$ et $V_{TP} = 0.617\ V$ sont des paramètres inhérents à la technologie utilisée. V_{DD} est la source de tension de 2.5 V.

Le but est de minimiser R_X et de maximiser f_{cli} en fonction de W_N, L_N, W_P et L_P, en respectant les conditions imposées par (5.10) et (5.11). Les limites technologiques imposent :

$$35.10^{-8} \leq L_N, L_P \leq 45.10^{-8} \tag{5.12}$$

$$1.10^{-6} \leq W_N, W_P \leq 30.10^{-6} \tag{5.13}$$

Les résultats optimaux sur les largeurs et longueurs des transistors NMOS et PMOS qui forment le convoyeur de courant sont obtenus en utilisant MO-TRIBES. Ces résultats sont présentés dans la Table 5.2. Ces résultats constituent le front de Pareto approché du problème de dimensionnement. La Figure 5.11 représente celui-ci dans le plan $(-f_{cli}/R_X)$, ainsi que les solutions non-dominées dans l'espace des solutions. Cette figure ne présente que les valeurs pour lesquelles f_{ci} est supérieure à 1 GHz. Le courant de polarisation I_0 est égal à 100 μA. Le critère d'arrêt pour MO-TRIBES est $Max_{Fes} = 10000$.

La Table 5.3 montre les bornes des fronts de Pareto obtenus en fonction du courant de polarisation I_0. On voit sur ce tableau que la fréquence de coupure augmente lorsque I_0 augmente, alors que la résistance d'entrée décroît lorsque I_0 augmente. On en déduit que plus I_0 est grand, plus le circuit possédera de bonnes performances. Cependant, sachant que produire des forts courants de polarisation est assez complexe, il est donc important de trouver un bon compromis entre la valeur du courant de polarisation et les performances souhaitées pour le circuit. La Figure 5.12 montre les différents fronts de Pareto, dans le plan $(-f_{cli}/R_X)$.

Les simulations sont réalisées en utilisant la technologie ADS 0.35 μm, une alimentation de 2.5 V et un courant de polarisation I_0 égal à 100 mA. Les Figures 5.13 et 5.14 présentent les simulations SPICE [1] obtenues à l'aide des valeurs de Table 5.3 pour $I_0 = 100\ \mu A$. Les résultats de simulation sont en accord avec les résultats théoriques calculés précédemment. La Table 5.4 illustre cette remarque.

1. SPICE (*Simulation Program with Integrated Circuit Emphasis*) est un logiciel de simulation généraliste de circuits électroniques analogiques.

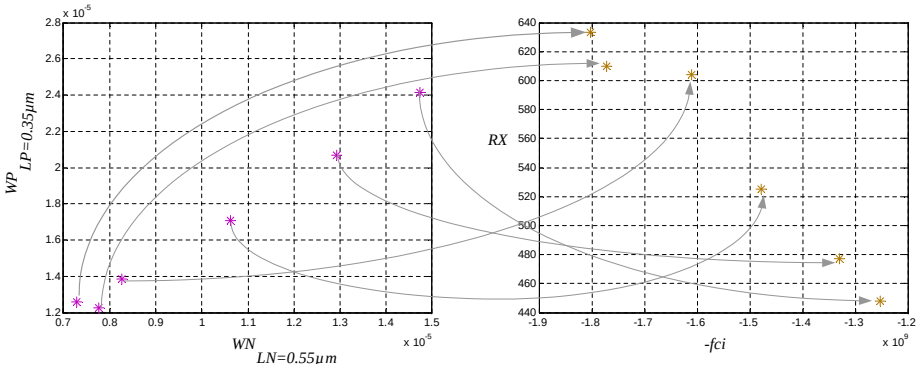


FIGURE 5.11 – Solutions non-dominées et front de Pareto du problème de dimensionnement.

TABLE 5.3 – Bornes du front de Pareto en fonction du courant de polarisation I_0.

I_0 (μA)	f_{ci-min} (GHz)	R_{X-min} (Ω)	f_{ci-max} (GHz)	R_{X-max} (Ω)
50	0.866	1376	0.027	714
100	1.802	633	0.059	382
150	1.721	435	0.078	336
200	2.027	338	0.090	285
250	1.940	272	0.097	249
300	2.042	230	0.107	224

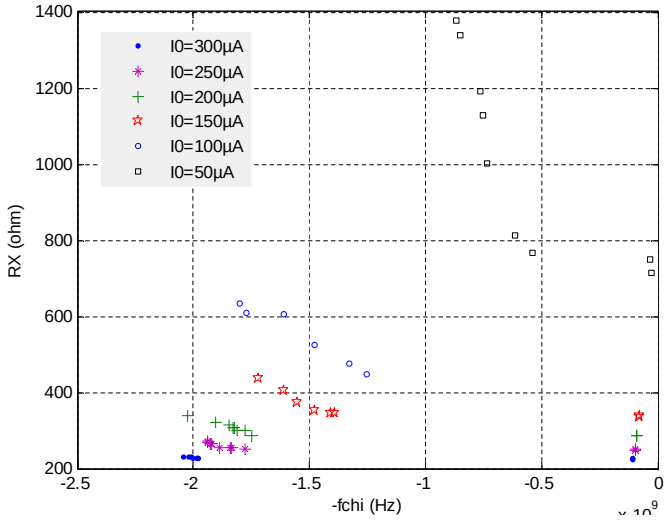
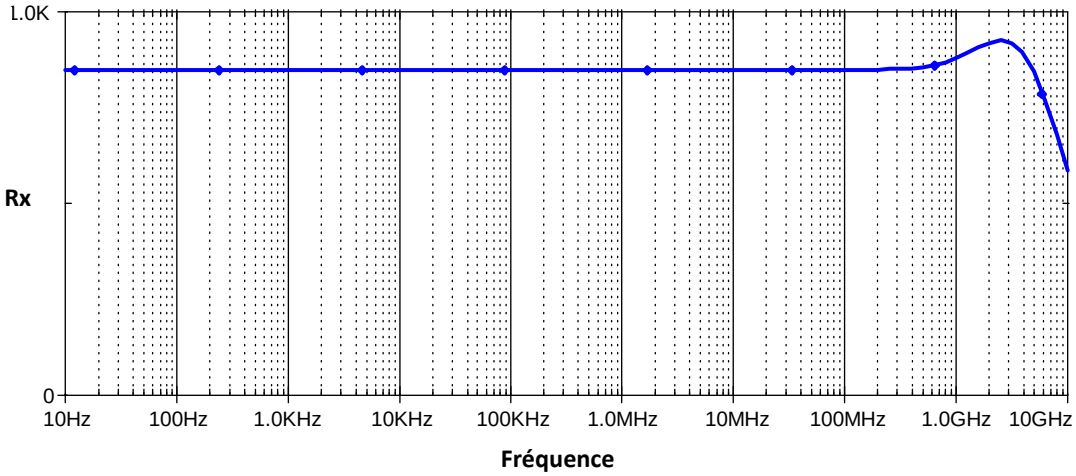
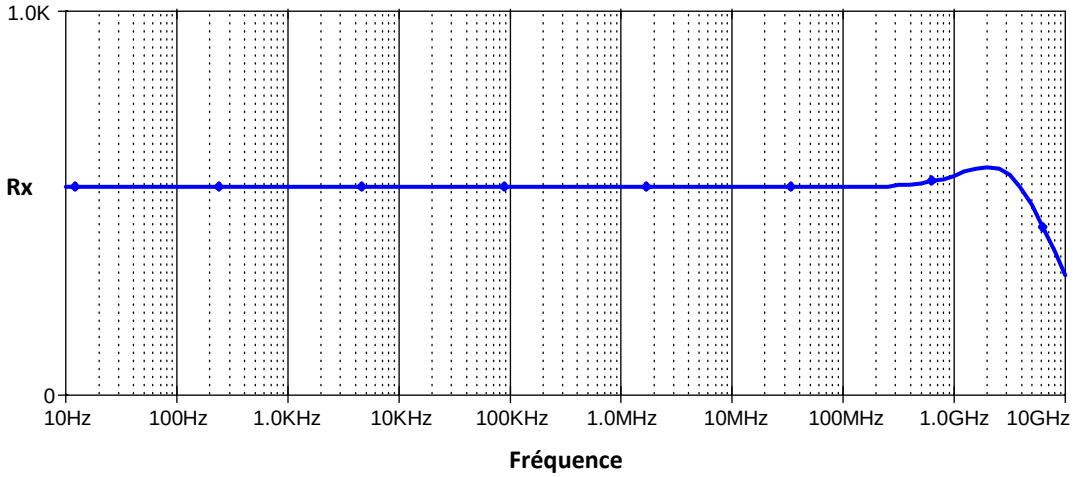


FIGURE 5.12 – Front de Pareto en fonction du courant de polarisation I_0.



(a) Solution donnant un f_{ci} maximum



(b) Solution donnant un R_X minimum

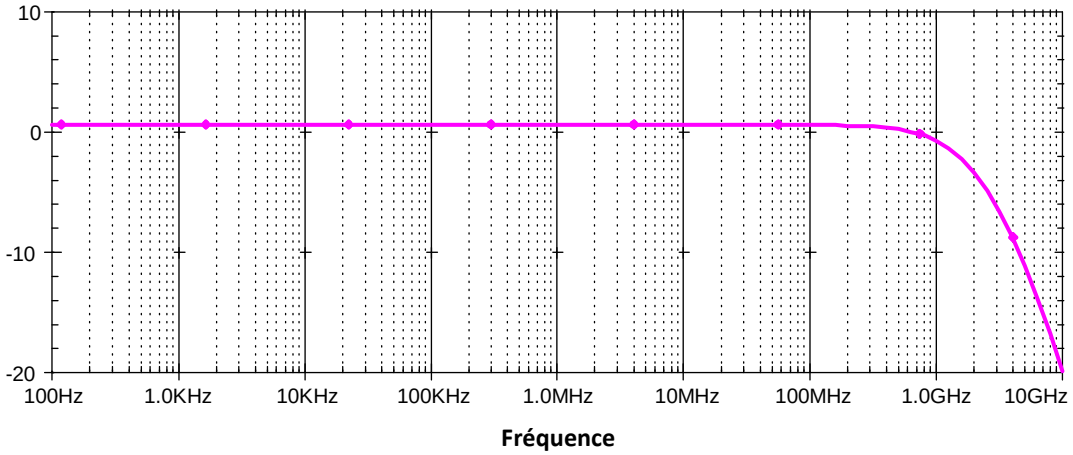
FIGURE 5.13 – Simulations SPICE pour R_X.

Ce convoyeur de courant a été optimisé dans [Ben Salem et al., 2006] en utilisant une approche agrégative. L'unique solution trouvée par Ben Salem et al. appartient au front de Pareto donné par MO-TRIBES et correspond à la borne inférieure (f_{ci-min}/R_{X-min}).

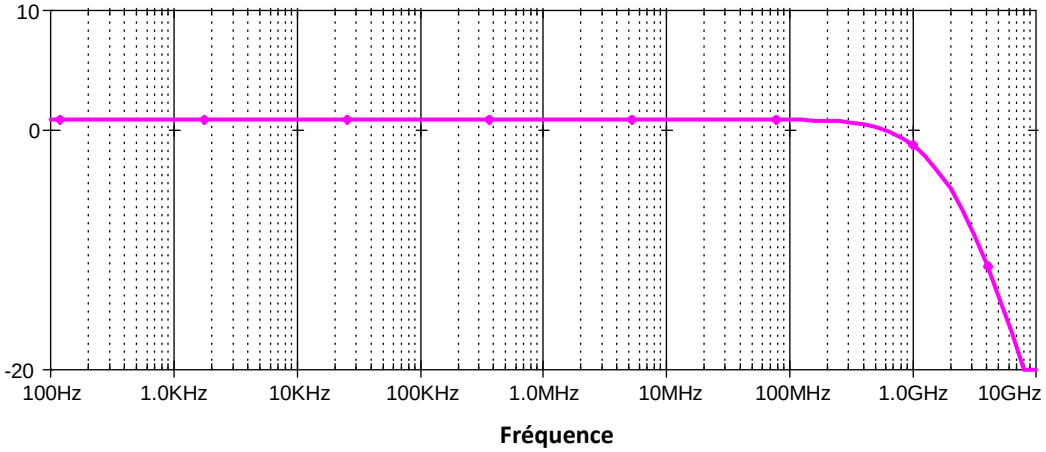
Ce travail a fait l'objet d'une publication [Cooren et al., 2007b].

5.4 Conclusion

Dans ce chapitre, nous avons présenté deux applications de l'optimisation au problème de dimensionnement de circuits électroniques analogiques. Ce type de problème est crucial lors de la conception de circuits analogiques, car les performances des circuits sont fortement dépendantes des dimensions des transistors. Il n'existe pas de méthode "automatique" pour résoudre ce genre de problème, la résolution reposant le plus souvent sur l'expérience d'un expert. Ceci implique que le problème de dimensionnement est un problème long et fastidieux à résoudre. Ce chapitre montre



(a) Solution donnant un f_{ci} maximum



(b) Solution donnant un R_X minimum

FIGURE 5.14 – Simulations SPICE pour f_{ci}.

TABLE 5.4 – Comparaison entre les résultats théoriques et les simulations.

	Résultats théoriques		Simulations	
	f_{ci} (GHz)	R_X (Ω)	f_{ci} (GHz)	R_X (Ω)
Borne inférieure du front de Pareto	1.253	448	1.211	566
Borne supérieure du front de Pareto	1.802	633	1.577	874

comment l'utilisation des métaheuristiques permet de trouver rapidement des solutions convenables en terme de performance des circuits.

Le premier problème, mono-objectif, a consisté à maximiser le gain d'un amplificateur de courant faible bruit. Le deuxième, multiobjectif, a consisté à simultanément minimiser la résistance d'entrée et maximiser la fréquence de coupure d'un convoyeur de courant de seconde génération. Ces problèmes ont été traités respectivement à l'aide de TRIBES et MO-TRIBES. Nous avons montré dans ce chapitre que les deux problèmes sont correctement résolus.

Conclusion et perspectives

Dans cette thèse, nous avons présenté un travail basé sur un algorithme d'Optimisation par Essaim Particulaire adaptative appelé TRIBES. L'optimisation par essaim particulaire est une métaheuristique destinée à la résolution de problèmes à variables continues, inspirée du comportement en essaim de certains animaux. Un essaim de particules présente un comportement auto-organisé qui conduit, par des interactions au niveau local, à l'émergence d'un comportement global complexe. Un des inconvénients de cette méthode est que son comportement est très fortement dépendant des valeurs données à ses différents paramètres de contrôle. Un temps non négligeable est donc nécessaire au préalable pour régler le jeu de paramètres en fonction du problème posé. Pour s'affranchir de cette étape fastidieuse et faciliter la prise en main de l'algorithme par un utilisateur novice, de nombreuses recherches ont été menées, afin de proposer des méthodes adaptatives. Une méthode adaptative est une méthode pour laquelle les valeurs des paramètres sont calculées au cours de l'exécution de l'algorithme, en fonction du comportement des particules, et non imposées par l'utilisateur.

TRIBES est un algorithme d'optimisation par essaim particulaire adaptatif qui a été proposé par Maurice Clerc. Cet algorithme présente la particularité d'être totalement adaptatif, l'utilisateur n'a qu'à définir son problème et son critère d'arrêt, sans se préoccuper des paramètres de l'algorithme. Tous les paramètres de contrôle sont calculés de manière autonome par l'algorithme, aucune intervention humaine n'étant nécessaire à la bonne résolution d'un problème. La première étape de ce travail de thèse a été de mener une étude des performances de TRIBES, afin d'en dégager les principaux défauts et, ainsi, de nouvelles pistes de développement. Cette étude a été menée en utilisant la procédure de test définie lors de la conférence *IEEE International Conference Evolutionary Computation 2005*, qui vise à proposer un standard de comparaison entre les différentes métaheuristiques. L'étude menée montre tout d'abord que, certes, TRIBES n'est pas encore le meilleur algorithme d'optimisation, mais il présente des résultats tout-à-fait compétitifs vis-à-vis de la majorité des algorithmes. Le principal constat est que TRIBES souffre d'un déficit de diversité au sein de l'essaim. En effet, l'algorithme converge vite vers un optimum, qui est souvent un optimum local, et n'arrive pas à s'en extraire. En d'autres termes, les mécanismes de TRIBES n'arrivent pas à maintenir la diversité nécessaire parmi les particules, afin d'explorer les différents optima locaux de la fonction objectif. Afin de résoudre ce problème, nous avons défini deux nouveaux mécanismes pour TRIBES : une initialisation régulière et une stratégie de déplacement. L'initialisation régulière a pour but d'initialiser les particules de manière à ce que, dès le début du traitement, l'espace de recherche soit exploré de la manière la plus large possible par l'essaim de particules. Le but est d'explorer l'espace de recherche aussi largement que possible. La nouvelle stratégie de déplacement est destinée aux particules qui présentent un comportement défaillant. Afin d'améliorer le comportement de ces particules, nous avons défini une nouvelle stratégie de déplacement, basée sur l'utilisation d'un algorithme à estimation de distribution. L'utilisation d'un tel algorithme permet d'avoir des déplacements plus larges, conditionnés par les meilleures particules de l'essaim et, ainsi, de diversifier le comportement d'une particule défaillante. Ces deux mécanismes aboutissent à une amélioration sensible des résultats de TRIBES.

TRIBES est destiné à la résolution de problèmes mono-objectif. Face au besoin croissant de méthodes multiobjectif, nous avons développé au cours de cette thèse une version multiobjectif de TRIBES, dénommée MO-TRIBES. Comme dans le cas de TRIBES, MO-TRIBES est un algorithme pour lequel la définition des paramètres de contrôle ne nécessite aucune intervention de l'utilisateur. MO-TRIBES est un algorithme multiobjectif qui utilise une approche basée sur la dominance, au sens de Pareto, pour chercher l'ensemble des solutions du problème posé. Tout au long du traitement, on stocke les solutions non-dominées au sein d'une archive externe, dont la taille et le processus de mise à jour sont définis automatiquement. La diversité au sein de celle-ci est maintenue grâce à un critère de maximisation de *distance d'encombrement*. De même, une certaine forme de diversité est maintenue au sein de l'essaim par l'intermédiaire de multiples réinitialisations de l'essaim. Le procédé d'information des particules étant différent de celui du cas mono-objectif, nous avons défini une nouvelle topologie du graphe d'information des particules. Ce graphe permet de favoriser les solutions non-dominées, afin d'obtenir la meilleure approximation possible du front de Pareto du problème posé. MO-TRIBES a été comparé à deux des algorithmes les plus performants, NSGA-II et MOPSO, et présente des résultats très encourageants, surtout si l'on considère que les pistes d'amélioration sont nombreuses.

Nos algorithmes ont été appliqués dans deux domaines différents. Tout d'abord, nous avons montré, dans le cas mono-objectif, que les métaheuristiques peuvent être une alternative rapide et performante au problème de seuillage d'images haute-résolution, telles que les images IRM. Ensuite, TRIBES et MO-TRIBES ont été utilisés pour résoudre des problèmes de dimensionnement de transistors en électronique analogique. Ce type de problème est crucial en électronique, car les performances des circuits sont très fortement dépendantes des dimensions des transistors. Peu de méthodes automatiques existent pour résoudre ce type de problème. Nous avons montré que, dans les cas mono-objectif et multiobjectif, l'utilisation de méthodes d'optimisation est ici aussi une alternative crédible, tant en termes de performance que de temps de calcul.

Les perspectives de ce travail sont nombreuses, tant du point de vue théorique que pratique. Nous avons montré dans cette thèse que les méthodes adaptatives ont un vrai avenir dans le monde de la recherche opérationnelle car elles permettent non seulement une prise en main très facile des algorithmes d'optimisation, mais aussi d'obtenir des résultats tout-à-fait compétitifs.

TRIBES est un algorithme modulaire. Il est donc facilement possible de lui ajouter de nouveaux mécanismes. Dans le cas de TRIBES, les deux points sujets à amélioration sont les règles d'adaptation structurelle et les stratégies de déplacement des particules. Ainsi, il serait bénéfique de définir de nouvelles stratégies de déplacement des particules, afin de proposer un déplacement "idéal" à la particule, en se basant sur son passé récent. Le but ici est de réaliser un apprentissage de l'espace de recherche afin de mieux guider les particules dans leur recherche. Par exemple, l'utilisation de fonctions d'interpolation, telles que les fonctions B-splines, pourrait permettre aux particules de mieux appréhender leurs déplacements dans l'espace de recherche. De même, il serait intéressant d'utiliser des stratégies de déplacement basées sur des paradigmes autres que celui de l'OEP, afin de pouvoir bénéficier des avantages de ces autres méthodes. Dans l'idéal, chaque type d'historique devrait posséder une stratégie de déplacement qui lui est propre. L'idée est que chaque stratégie de déplacement apporte une solution au problème posé par les déplacements antérieurs de la particule.

L'étape de mise à jour des liens d'information à l'intérieur de TRIBES est aussi un enjeu important, car une bonne gestion de la structure de l'essaim pourrait permettre de maintenir la diversité nécessaire au sein de l'essaim. La question est ici de savoir quelle méthode utiliser pour modifier de manière optimale la structure de l'essaim, mais aussi de savoir quel est le moment le plus opportun pour réaliser ce changement. En ce qui concerne l'instant de modification, il est clair qu'il est néces-

saire de définir un critère de "stagnation" de l'essaim qui permette de mesurer la capacité de l'essaim à pouvoir encore améliorer sa performance ou, à l'opposé, de mesurer la tendance de l'essaim à être bloqué dans une situation quasi-figée. Il serait peut-être nécessaire de revoir la structure du graphe d'information de TRIBES en testant les effets d'autres topologies (par exemple, topologie circulaire sur les chamans) sur les performances de l'essaim. Ceci implique aussi une mise à jour des liens d'information différente lors de l'ajout/suppression de particules. De plus, les statuts utilisés pour les opérations de suppression sont assez binaires, il serait intéressant de créer des statuts intermédiaires qui rendent mieux comptent de la situation d'une tribu ou d'une particule au sein de l'essaim.

Les remarques précédentes ont pour but d'améliorer TRIBES en conférant aux particules un meilleur pouvoir d'apprentissage de l'espace de recherche tout en maintenant au sein de l'essaim une forte diversité. Ce genre de caractéristique est aussi appréciable dans le cas multiobjectif, donc les propositions de perspectives pour TRIBES peuvent aussi servir de perspectives pour MO-TRIBES. De plus, dans le cas multiobjectif, il est possible d'améliorer la gestion de l'archive. Par exemple, la taille de celle-ci est adaptée suivant une règle définie empiriquement. Il serait intéressant de mener une étude plus approfondie, afin de déterminer une règle optimale pour définir la taille de l'archive. De même, le choix des meilleurs informateurs au sein de l'archive pourrait favoriser les particules qui ont la plus grande *distance d'encombrement*. D'autres critères d'évaluation des solutions non-dominées peuvent être utilisés.

Considérant que TRIBES converge très rapidement vers un optimum local de la fonction objectif, il serait intéressant de modifier TRIBES afin de l'exploiter en optimisation dynamique. En effet, l'utilisation de TRIBES pourrait permettre de détecter rapidement l'optimum global de la fonction entre deux modifications de la topologie de celle-ci.

Annexe A : Principales fonctions de test

Principales fonctions de test mono-objectif

Toutes les données nécessaires au calcul de ces fonctions (matrices de rotation, vecteurs de translation, code sources,...) sont disponibles à l'adresse suivante : `http://www3.ntu.edu.sg/home/EPNSugan/`

Fonction Sphère (F1)

$$F_1(\overrightarrow{x}) = \sum_{i=1}^{D} z_i^2 + f_{biais1},\ f_{biais1} = -450$$

$$\overrightarrow{z} = \overrightarrow{x} - \overrightarrow{o},\quad \overrightarrow{x} = [x_1, \ldots, x_D] \in [-100, 100]^D$$

$$\overrightarrow{o} = [o_1, \ldots, o_D], \text{ l'optimum global}$$

Fonction Schwefel 1.2 (F2)

$$F_2(\overrightarrow{x}) = \sum_{i=1}^{D} \left(\sum_{j=1}^{i} z_j^2 \right) + f_{biais2},\ f_{biais2} = -450$$

$$\overrightarrow{z} = \overrightarrow{x} - \overrightarrow{o},\quad \overrightarrow{x} = [x_1, \ldots, x_D] \in [-100, 100]^D$$

$$\overrightarrow{o} = [o_1, \ldots, o_D], \text{ l'optimum global}$$

Fonction Elliptic (F3)

$$F_3(\overrightarrow{x}) = \sum_{i=1}^{D} 10^{6 \cdot \frac{i-1}{D-1}} z_i^2 + f_{biais3},\ f_{biais3} = -450$$

$$\overrightarrow{z} = (\overrightarrow{x} - \overrightarrow{o}) \cdot M,\quad \overrightarrow{x} = [x_1, \ldots, x_D] \in [-100, 100]^D$$

$$\overrightarrow{o} = [o_1, \ldots, o_D], \text{ l'optimum global}$$

M est une matrice de rotation orthogonale

Fonction Schwefel 1.2 bruitée (F4)

$$F_4(\overrightarrow{x}) = \left(\sum_{i=1}^{D}\left(\sum_{j=1}^{i} z_j^2\right)\right)\cdot(1+0.4\cdot N(0,1)) + f_{biais4},\ f_{biais4} = -450$$

$$\overrightarrow{z} = \overrightarrow{x} - \overrightarrow{o},\quad \overrightarrow{x} = [x_1,\ldots,x_D] \in [-100,100]^D$$

$\overrightarrow{o} = [o_1,\ldots,o_D]$, l'optimum global

Fonction Schwefel 2.6 bruitée (F5)

$$F_5(\overrightarrow{x}) = \max_{i\in\{1,\ldots,D\}}\{|A_i\overrightarrow{x} - B_i|\} + f_{biais5},\quad f_{biais5} = -310$$

A est un matrice DxD, A_i est la i^{eme} ligne de A, $\det(A) \neq 0$

$$B_i = A_i\cdot\overrightarrow{o},\ o_i = U(-100,100),\ i = \{1,\ldots,D\}$$

$$\overrightarrow{z} = \overrightarrow{x} - \overrightarrow{o},\quad \overrightarrow{x} = [x_1,\ldots,x_D] \in [-100,100]^D$$

$\overrightarrow{o} = [o_1,\ldots,o_D]$, l'optimum global

Fonction Rosenbrock (F6)

$$F_6(\overrightarrow{x}) = \sum_{i=1}^{D-1}\left(100\left(z_i^2 - z_{i+1}\right)^2 + (z_i - 1)^2\right) + f_{biais6},\ f_{biais6} = 390$$

$$\overrightarrow{z} = \overrightarrow{x} - \overrightarrow{o} + 1,\quad \overrightarrow{x} = [x_1,\ldots,x_D] \in [-100,100]^D$$

$\overrightarrow{o} = [o_1,\ldots,o_D]$, l'optimum global

Fonction Griewank (F7)

$$F_7(\overrightarrow{x}) = \sum_{i=1}^{D}\frac{z_i^2}{4000} - \prod_{i=1}^{D}\cos\left(\frac{z_i}{\sqrt{i}}\right) + 1 + f_{biais7},\ f_{biais7} = -180$$

$$\overrightarrow{z} = (\overrightarrow{x} - \overrightarrow{o})\cdot M,\quad \overrightarrow{x} = [x_1,\ldots,x_D] \in [0,600]^D$$

$M = M'\cdot(1+0.3\cdot|N(0,1)|)$, M' est une matrice de transformation linéaire dont *nombre de contionnement* $= 3$ [2]

$\overrightarrow{o} = [o_1,\ldots,o_D]$, l'optimum global

2. Le *nombre de conditionnement* d'une matrice A mesure la complexité de son calcul numérique et est égal à $\|A^{-1}\|\cdot\|A\|$

Fonction Ackley (F8)

$$F_8\left(\overrightarrow{x}\right) = -20 \cdot e^{-0.2 \cdot \sqrt{\frac{1}{D} \cdot \sum_{i=1}^{D} z_i^2}} - e^{\frac{1}{D} \cdot \sum_{i=1}^{D} \cos(2\pi z_i)} + 20 + e + f_{biais8},\ f_{biais8} = -140$$

$$\overrightarrow{z} = (\overrightarrow{x} - \overrightarrow{o}) \cdot M, \quad \overrightarrow{x} = [x_1, \ldots, x_D] \in [-32, 32]^D$$

M est une matrice de transformation linéaire dont *nombre de conditionnement* $= 100$

$$\overrightarrow{o} = [o_1, \ldots, o_D], \text{ l'optimum global}$$

$$o_{2j-1} = -32 \cdot o_{2j},\ j = \left\{1, 2, \ldots, \left\lfloor \frac{D}{2} \right\rfloor\right\}$$

Fonction Rastrigin (F9)

$$F_9\left(\overrightarrow{x}\right) = \sum_{i=1}^{D} \left(z_i^2 + 10 \cdot \cos\left(2\pi z_i\right) + 10\right) + f_{biais9},\ f_{biais9} = -330$$

$$\overrightarrow{z} = \overrightarrow{x} - \overrightarrow{o}, \quad \overrightarrow{x} = [x_1, \ldots, x_D] \in [-5, 5]^D$$

$$\overrightarrow{o} = [o_1, \ldots, o_D], \text{ l'optimum global}$$

Fonction Rastrigin pivotée (F10)

$$F_{10}\left(\overrightarrow{x}\right) = \sum_{i=1}^{D} \left(z_i^2 + 10 \cdot \cos\left(2\pi z_i\right) + 10\right) + f_{biais10},\ f_{biais10} = -330$$

$$\overrightarrow{z} = (\overrightarrow{x} - \overrightarrow{o}) \cdot M, \quad \overrightarrow{x} = [x_1, \ldots, x_D] \in [-5, 5]^D$$

M est une matrice de transformation linéaire dont *nombre de conditionnement* $= 2$

$$\overrightarrow{o} = [o_1, \ldots, o_D], \text{ l'optimum global}$$

Fonction Weierstrass (F11)

$$F_{11}\left(\overrightarrow{x}\right) = \sum_{i=1}^{D} \left(\sum_{k=0}^{k_{max}} a^k \cos\left(2\pi b^k \left(z_i + 0.5\right)\right) \right) - D \cdot \sum_{k=0}^{k_{max}} \left(a^k \cos\left(2\pi b^k \cdot 0.5\right)\right) + f_{biais11},\ f_{biais11} = 90$$

$$\overrightarrow{z} = (\overrightarrow{x} - \overrightarrow{o}) \cdot M, \quad \overrightarrow{x} = [x_1, \ldots, x_D] \in [-0.5, 0.5]^D,\ a = 0.3,\ b = 3,\ k_{max} = 20$$

M est une matrice de transformation linéaire dont *nombre de conditionnement* $= 5$

$$\overrightarrow{o} = [o_1, \ldots, o_D], \text{ l'optimum global}$$

Fonction Schwefel 2.13 (F12)

$$F_{12}(\overrightarrow{x}) = \sum_{i=1}^{D} (A_i - B_i(\overrightarrow{x}))^2 + f_{biais12},\ f_{biais12} = -460,\quad \overrightarrow{x} = [x_1, \ldots, x_D] \in [-\pi, \pi]^D$$

$$A_i = \sum_{i=1}^{D} (a_{ij} \sin(\alpha_j) + b_{ij} \cos(\alpha_j)),\ B_i = \sum_{i=1}^{D} (a_{ij} \sin(x_j) + b_{ij} \cos(x_j))$$

$$a_{ij}, b_{ij} \in [-100, 100],\ \alpha_j \in [-\pi, \pi]$$

Principales fonctions de test multiobjectif

TABLE A.1 - Fonctions de test multiobjectif.

D	k	Nom	Espace de recherche	Expression analytique	Référence
2	2	Deb	$[0.1, 1]^2$	$f_1(\overrightarrow{x}) = x_1$ $f_2(\overrightarrow{x}) = \frac{2 - e^{-\left(\frac{x_2 - 0.2}{0.004}\right)^2} - 0.8 \cdot e^{-\left(\frac{x_2 - 0.6}{0.4}\right)^2}}{x_1}$	[Deb, 1999]
30	2	ZDT1	$[0, 1]^{30}$	$f_1(\overrightarrow{x}) = x_1$ $g(\overrightarrow{x}) = 1 + 9 \sum_{i=2}^{D} \frac{x_i}{D-1}$ $h(\overrightarrow{x}, f_1, g) = 1 - \sqrt{\frac{f_1(\overrightarrow{x})}{g(\overrightarrow{x})}}$ $f_2(\overrightarrow{x}) = g(\overrightarrow{x}) \cdot h(\overrightarrow{x}, f_1, g)$	[Zitzler et al., 2000]
30	2	ZDT2	$[0, 1]^{30}$	$f_1(\overrightarrow{x}) = x_1$ $g(\overrightarrow{x}) = 1 + 9 \sum_{i=2}^{D} \frac{x_i}{D-1}$ $h(\overrightarrow{x}, f_1, g) = 1 - \left(\frac{f_1(\overrightarrow{x})}{g(\overrightarrow{x})}\right)^2$ $f_2(\overrightarrow{x}) = g(\overrightarrow{x}) \cdot h(\overrightarrow{x}, f_1, g)$	[Zitzler et al., 2000]
30	2	ZDT3	$[0, 1]^{30}$	$f_1(\overrightarrow{x}) = x_1$ $g(\overrightarrow{x}) = 1 + 9 \sum_{i=2}^{D} \frac{x_i}{D-1}$ $h(\overrightarrow{x}, f_1, g) = 1 - \left(\frac{f_1(\overrightarrow{x})}{g(\overrightarrow{x})}\right) \cdot sin(10\pi f_1(\overrightarrow{x})) - \sqrt{\frac{f_1(\overrightarrow{x})}{g(\overrightarrow{x})}}$ $f_2(\overrightarrow{x}) = g(\overrightarrow{x}) \cdot h(\overrightarrow{x}, f_1, g)$	[Zitzler et al., 2000]
10	2	ZDT6	$[0, 1]^{10}$	$f_1(\overrightarrow{x}) = 1 - e^{-4x_1} \sin^6(6\pi x_1)$ $g(\overrightarrow{x}) = 1 + 9 \left(\sum_{i=2}^{D} \frac{x_i}{D-1}\right)^{0.25}$ $h(\overrightarrow{x}, f_1, g) = 1 - \left(\frac{f_1(\overrightarrow{x})}{g(\overrightarrow{x})}\right)^2$ $f_2(\overrightarrow{x}) = g(\overrightarrow{x}) \cdot h(\overrightarrow{x}, f_1, g)$	[Zitzler et al., 2000]

D	k	**Nom**	**Espace de recherche**	**Expression analytique**	**Référence**
2	3	MOP5	$[-30,30]^2$	$f_1(\overrightarrow{x}) = 0.5\left(x_1^2+x_2^2\right)+\sin\left(x_1^2+x_2^2\right)$ $f_2(\overrightarrow{x}) = \frac{(3x_1-2x_2+4)^2}{8}+\frac{(x_1-x_2+1)^2}{27}+15$ $f_3(\overrightarrow{x}) = \frac{1}{x_1^2+x_2^2+1}-1.1\cdot e^{-x_1^2-x_2^2}$	[Coello Coello et al., 2002]
2	2	MOP6	$[0,1]^2$	$f_1(\overrightarrow{x}) = x_1$ $f_2(\overrightarrow{x}) = (1+10x_2)\left(1-\left(\frac{x_1}{1+10x_2}\right)^2-\left(\frac{x_1}{1+10x_2}\right)\sin(2\pi 4x_1)\right)$	[Coello Coello et al., 2002]

Annexe B : Résultats numériques de TRIBES

TABLE B.1 – Erreur pour les fonctions unimodales en dimension 10 pour FEs=10^3, 10^4 et 10^5 .

FEs		F1	F2	F3	F4	F5
FEs=10^3	1^{er}	0.004347	17.156654	23736.314425	39.090571	146.139053
	$7^{ème}$	0.011044	37.389443	44778.393803	133.988893	598.534137
	$13^{ème}$	0.03552	58.212032	130599.062043	237.769180	862.570764
	$19^{ème}$	0.096339	139.37427	318787.308758	596.315063	1313.068874
	$25^{ème}$	0.276081	346.374378	1594493.870254	3932.029550	3228.512443
	Moyenne	0.068943	100.607688	273638.075237	577.302159	1110.369922
	Variance	0.075044	91.541188	388777.309094	834.977069	781.104211
FEs=10^4	1^{er}	0	1.55e-10	4.001777e-11	0.000004	0.095210
	$7^{ème}$	0	3.36e-09	7.671247e-09	0.000085	1.716032
	$13^{ème}$	0	6.61e-08	5.475368e-07	0.000957	11.366620
	$19^{ème}$	0	7.61e-06	0.000026	0.009582	45.927941
	$25^{ème}$	0	2.01e-03	0.002139	0.363307	1119.663894
	Moyenne	0	1.57e-04	0.000117	0.025930	72.416017
	Variance	0	4.40e-04	0.00042	0.072792	217.650700
FEs=10^5	1^{er}	0	0	0	0	0
	$7^{ème}$	0	0	0	0	0
	$13^{ème}$	0	0	0	0	5.68e-13
	$19^{ème}$	0	0	0	0	2.84e-08
	$25^{ème}$	0	5.68e-14	0	3.64e-14	1.11e-05
	Moyenne	0	2.27e-15	0	9.09e-15	9.18e-07
	Variance	0	1.11e-14	0	2.08e-14	2.44e-06

TABLE B.2 – Erreur pour les fonctions multimodales en dimension 10 pour FEs=10^3, 10^4 et 10^5 .

FEs		F6	F7	F8	F9	F10	F11	F12
FEs=10^3	1^{er}	203.463571	0.623361	20.485670	15.10467	16.23971	3.673786	1086.721513
	$7^{ème}$	1646.790313	0.982154	20.682525	33.675252	33.74019	6.84937	2087.954288
	$13^{ème}$	2127.361491	1.05142	20.769963	44.956646	47.748059	8.270853	5392.676747
	$19^{ème}$	6475.698481	1.462515	20.814313	55.474189	63.569776	8.761013	14089.042494
	$25^{ème}$	38601.955678	5.384111	20.915608	76.821327	84.542945	10.9190	42976.859515
	Moyenne	6586.931554	1.33978	20.745941	43.309673	48.262753	7.935141	9990.359131
	Variance	9386.941213	0.902277	0.096755	16.000977	17.973327	1.569063	10625.813191
FEs=10^4	1^{er}	0.014145	0.061571	20.416319	2.984879	5.969754	2.793658	1.822694
	$7^{ème}$	4.386041	0.115684	20.485168	8.954626	10.94457	4.702974	150.871807
	$13^{ème}$	5.453284	0.137799	20.525583	14.924371	17.909248	5.44391	1576.117425
	$19^{ème}$	7.075625	0.221471	20.600516	21.889039	23.878972	6.58871	5735.460084
	$25^{ème}$	30.227322	0.440146	20.764823	50.742348	41.788129	8.413328	14058.362618
	Moyenne	6.385858	0.180578	20.545279	16.371514	19.052126	5.567022	3252.791878
	Variance	5.564184	0.10215	0.088329	11.22564	8.864428	1.424877	3782.702898
FEs=10^5	1^{er}	0.004314	0.015287	20.416319	2.984877	5.969754	2.792143	0.009991
	$7^{ème}$	0.541085	0.051984	20.485168	5.969754	9.098987	3.456094	3.347386
	$13^{ème}$	0.747254	0.073769	20.525583	7.959667	10.94454	4.158164	18.856007
	$19^{ème}$	1.103226	0.091044	20.600516	9.949586	12.934458	4.871277	208.277865
	$25^{ème}$	2.169582	0.201614	20.764823	16.914269	27.858783	6.097017	5184.562815
	Moyenne	0.85882	0.077474	20.545279	8.556642	12.118002	4.233992	401.055407
	Variance	0.570887	0.03942	0.088329	3.66922	4.660575	0.958315	1066.232340

TABLE B.3 – Erreur pour les fonctions étendues et composées en dimension 10 pour FEs=10^3, 10^4 et 10^5 .

FEs		F13	F14	F15	F16	F17
FEs=10^3	1^{er}	3.171010	3.475790	197.489562	189.184315	206.397449
	$7^{ème}$	4.066271	4.002326	423.264014	235.981761	285.067996
	$13^{ème}$	4.812523	4.150536	470.853547	296.892300	371.981281
	$19^{ème}$	5.322586	4.300356	626.471829	312.520238	477.090661
	$25^{ème}$	6.044543	4.508500	777.774774	556.601713	789.774458
	Moyenne	4.703398	4.138838	491.856725	307.959855	426.898085
	Variance	0.749118	0.239739	143.137483	96.034311	180.843460
FEs=10^4	1^{er}	0.083393	2.798398	124.185470	155.201434	163.389636
	$7^{ème}$	0.701463	3.269073	228.681273	188.317715	179.965528
	$13^{ème}$	1.156255	3.576401	384.060458	204.072135	202.972766
	$19^{ème}$	1.434762	3.695722	398.763970	218.378360	263.948551
	$25^{ème}$	1.767060	4.022634	646.622212	553.128678	707.856708
	Moyenne	1.084210	3.488787	346.902015	228.777700	234.821253
	Variance	0.436958	0.325614	132.223074	95.73212	105.063253
FEs=10^5	1^{er}	0.348796	2.142077	92.820102	145.067276	149.27214
	$7^{ème}$	0.472609	2.58856	126.050968	169.775478	163.169731
	$13^{ème}$	0.593114	2.752672	187.286285	176.438753	169.735632
	$19^{ème}$	0.674788	3.087005	264.809332	187.096185	186.123329
	$25^{ème}$	0.970072	3.510396	457.359199	215.349429	236.277438
	Moyenne	0.545902	2.777861	212.15829	177.530554	176.305663
	Variance	0.222001	0.341739	100.776783	17.13220	21.653134

TABLE B.4 – Probabilité que l'erreur moyenne donnée par TRIBES pour FEs= 10^3 soit plus petite que les erreurs moyennes données par les autres algorithmes.

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11
G-CMA-ES	24.72%	34.05%	85.06%	73.17%	8.30%	57.32%	30.30%	65.17%	29.57%	39.17%	50.51%
EDA	99.98%	96.97%	99.77%	99.66%	99.99%	90.29%	100%	100%	97.73%	99.13%	98.72%
DE	99.49%	94.62%	97.54%	96.22%	99.84%	90.28%	98.09%	63.19%	23.86%	98.1%	98.33%
L-CMA-ES	64.28%	64.99%	99.77%	99.99%	27.22%	41.19%	46.66%	39.51%	91.7%	91.74%	82.65%
BLX-GL50	100%	99.66%	99.64%	99.57%	99.99%	97.17%	100%	39.80%	93.69%	97.79%	98.32%
DMS-L-PSO	84.68%	99.57%	93.27%	99.97%	99.99%	72.82%	100%	39.63%	74.88%	92.16%	86.88%
L-SaDE	99.88%	99.75%	96.53%	98.98%	99.99%	94.6%	99.73%	44.23%	73.16%	91.79%	97.07%
SPC-PNX	79.01%	97.24%	97.19%	92.81%	99.99%	64.01%	62.94%	99.94%	94.88%	96.72%	95.78%
CoEVO	97.56%	98.99%	98.12%	99.39%	99.99%	94.67%	97.15%	24.98%	88.27%	94.69%	95.37%
K-PCX	0%	0.01%	79.90%	99.98%	99.99%	31.01%	36.52%	38.68%	99.15%	99.42%	39.89%
BLX-MA	100%	100%	99.77%	99.99%	99.99%	96.45%	97.44%	0%	31.26%	55.4%	68.92%
Standard PSO	100%	87.31%	32.81%	61.69%	50.79%	69.97%	82.36%	74.23%	47.33%	62.04%	86.80%

TABLE B.5 – Probabilité que l'erreur moyenne donnée par TRIBES pour FEs= 10^4 soit plus petite que les erreurs moyennes données par les autres algorithmes.

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11
G-CMA-ES	100%	35.78%	39.31%	36.36%	36.94%	54.02%	4.06%	35.62%	18.68%	10.29%	32.56%
EDA	100%	99.66%	99.87%	99.73%	93.9%	96.23%	99.93%	45.38%	99.52%	99.03%	99.85%
DE	100%	92.45%	99.42%	77.48%	37.28%	70.93%	99.98%	37.93%	0%	98.92%	99.78%
L-CMA-ES	100%	0%	39.52%	88.03%	0%	17.62%	4.3%	29.58%	95.06%	95.56%	81.51%
BLX-GL50	100%	100%	96.36%	99.71%	37.56%	79.11%	100%	38.59%	34.32%	93.27%	99.68%
DMS-L-PSO	100%	100%	99.87%	99.6%	46.45%	89.25%	99.87%	8.53%	20.82%	67.08%	87.15%
L-SaDE	100%	98.45%	96.75%	97.16%	71.78%	83.33%	100%	38.10%	19.37%	89.75%	91.93%
SPC-PNX	100%	79.33%	91.26%	77.66%	41.68%	69.38%	89.3%	100%	89.78%	98.59%	44.09%
CoEVO	100%	89.13%	90.99%	86.66%	99.68%	89.45%	99.87%	16.07%	89.71%	90.18%	98.91%
K-PCX	-	0%	87.73%	98.96%	99.94%	15.1%	58.48%	0%	98.58%	99.83%	73.32%
BLX-MA	100%	100%	99.25%	82.34%	55.59%	55.09%	99.39%	0.05%	16.23%	12.48%	89.06%
Standard PSO	-	0.29%	-	36.46%	52.87%	69.84%	51.08%	54.12%	18.64%	39.13%	66.54%

TABLE B.6 – Probabilité que l'erreur moyenne donnée par TRIBES pour FEs= 10^5 soit plus petite que les erreurs moyennes données par les autres algorithmes.

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11
G-CMA-ES	-	-	-	-	35.23%	6.53%	2.49%	0%	1.22%	0.50%	0.60%
EDA	-	-	-	-	100%	0%	100%	2.64%	100%	99.96%	97.5%
DE	-	-	-	-	-	0%	99.99%	10.65%	0%	99.99%	53.71%
L-CMA-ES		-		-	-	0%	0%	0%	99.51%	97.95%	38.03%
BLX-GL50	-	-	-	-	-	0%	5.07%	5.48%	2.43%	9.38%	15.49%
DMS-L-PSO	-	-	-	-	-	6.92%	26.38%	0%	0%	3.65%	61.9%
L-SaDE	-	-	-	-	100%	6.65%	7.91%	0%	0%	7.47%	63.23%
SPC-PNX	-	-	-	-	-	67.62%	57.62%	100%	14.65%	24.56%	6.12%
CoEVO	-	-	-	-	100%	90.16%	16.89%	0%	92.01%	93.2%	99.97%
K-PCX	-	-	-	-	100%	39.81%	76.17%	0%	0.53%	0.59%	89.63%
BLX-MA	-	-	-	-	100%	86.73%	100%	0%	1.35%	8.30%	61.64%
Standard PSO	-	-	-	-	100%	63.53%	51.87%	43.24%	14.41%	29.89%	59.41%

TABLE B.7 – Nombre d'évaluations de la fonction objectif nécessaires pour atteindre une précision donnée pour D=10.

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11
Précision	1E-06	1E-06	1E-06	1E-06	1E-06	1E-02	1E-02	1E-02	1E-02	1E-02	1E-02
1^{er}	1132	4407	7590	7473	19310	57732	14211	1E+05	1E+05	1E+05	1E+05
7^{me}	1341	5901	9445	14178	33030	1E+05	1E+05	1E+05	1E+05	1E+05	1E+05
13^{me}	1379	6405	10765	17679	54219	1E+05	1E+05	1E+05	1E+05	1E+05	1E+05
19^{me}	1405	7024	11535	18912	68466	1E+05	1E+05	1E+05	1E+05	1E+05	1E+05
25^{me}	1538	10990	12302	37259	100000	1E+05	1E+05	1E+05	1E+05	1E+05	1E+05
Moyenne	1364.72	6620.92	10422.96	17399.32	55448	98309.28	96568.44	1E+05	1E+05	1E+05	1E+05
Variance	82.27	1440.33	1368.08	5288.60	21856.65	8282.80	16811.14	0	0	0	0
Taux de succès	1	1	1	1	0.88	0.04	0.04	0	0	0	0
Taux de performance	1364.72	6620.92	10422.96	17399.32	63009.09	2457725	355275	-	-	-	-

TABLE B.8 – Comparaisons pour le test n°2. Comparaison des *Taux de succès* et des *Taux de Performance normalisés* pour les fonctions unimodales en dimension 10.

			F1	F2	F3	F4	F5
	Nombre de problèmes résolus	Taux de succès	1000	2400	6500	2900	5900
G-CMA-ES	5	100%	1.6 (25)	1 (25)	1 (25)	1 (25)	1 (25)
EDA	5	98%	10 (25)	4.6 (25)	2.5 (23)	4.1 (25)	4.2 (25)
DE	5	96%	29 (25)	19.2 (25)	18.5 (20)	17.9 (25)	6.9 (25)
Standard PSO	5	86%	1.92(25)	1.89(25)	1.28(25)	4.05 (24)	9.51(9)
L-CMA-ES	5	86%	1.7 (25)	1.1 (25)	1 (25)	65.5 (7)	1 (25)
BLX-GL50	4	80%	19 (25)	17.1 (25)	-	14.5 (25)	4.7 (25)
SPC-PNX	4	80%	6.7 (25)	12.9 (25)	-	10.7 (25)	6.8 (25)
CoEVO	4	80%	23 (25)	11.3 (25)	6.8 (25)	16.2 (25)	-
DMS-L-PSO	4	76%	12 (25)	5 (25)	1.8 (25)	-	18.6 (20)
L-SaDE	4	72%	10 (25)	4.2 (25)	8 (16)	15.9 (24)	-
BLX-MA	3	59%	12 (25)	15.4 (25)	-	25.9 (24)	-
K-PCX	3	57%	1 (25)	1 (25)	-	19.7 (21)	-
TRIBES	**5**	**98%**	**1.3 (25)**	**2.75 (25)**	**1.60(25)**	**5.99 (25)**	**10.67 (22)**

TABLE B.9 – Comparaisons pour le test n°2. Comparaison des *Taux de succès* et des *Taux de Performance normalisés* pour les fonctions multimodales en dimension 10.

			F6	F7	F8	F9	F10	F11
	Nombre de problèmes résolus	Taux de succès	7100	4700	59585	17000	55000	190000
G-CMA-ES	5	65%	1.5 (25)	1 (25)	-	4.5 (19)	1.2 (23)	1.4 (6)
K-PCX	3	45%	9.6 (22)	-	-	2.9 (24)	1 (22)	-
L-SaDE	3	37%	6.6 (24)	36.2 (6)	-	1 (25)	-	-
DMS-L-PSO	3	36%	1.3 (25)	126 (4)	-	2.1 (25)	-	-
DE	4	33%	7.3 (25)	255 (2)	-	10.6 (11)	-	1 (12)
L-CMA-ES	1	33%	7.7 (25)	1.2 (25)	-	-	-	-
BLX-GL50	3	25%	6.69 (25)	12.3 (9)	-	10 (3)	-	-
SPC-PNX	3	16%	1 (22)	383 (1)	-	-	-	5.8 (1)
Standard PSO	1	14%	8.22(21)	-	-	-	-	-
BLX-MA	1	12%	-	-	-	5.7 (18)	-	-
EDA	2	3%	-	404 (1)	-	-	-	2.9 (3)
CoEVO	0	0%	-	-	-	-	-	-
TRIBES	**2**	**1%**	**346.15(1)**	**75.59 (1)**	**-**	**-**	**-**	**-**

TABLE B.10 – Erreurs obtenues par TRIBES pour les fonctions unimodales F1 à F5 pour FEs=10^3, 10^4, 10^5 et $3 \cdot 10^5$ en dimension 30.

FEs		F1	F2	F3	F4	F5
FEs=10^3	1^{er}	545.401319	23753.455258	4.27361e+07	32573.551403	7055.819362
	$7^{ème}$	763.736762	36551.200224	8.51668e+07	45090.823064	10868.133323
	$13^{ème}$	952.377579	41729.732379	9.91835e+07	57324.701513	12457.704085
	$19^{ème}$	1175.006339	48020.937603	1.23047e+08	77243.382507	13882.372397
	$25^{ème}$	2135.289475	73057.951110	2.30009e+08	107561.251077	17034.082175
	Moyenne	1078.901830	42257.908075	1.08777e+08	61096.008923	12098.970689
	Variance	438.636560	10513.104038	4.22558e+07	19032.203746	2462.033587
FEs=10^4	1^{er}	8.76923e-10	1157.692640	3.60156e+06	8932.572469	3187.995258
	$7^{ème}$	2.34268e-08	2456.192247	7.75657e+06	14849.919239	4014.106435
	$13^{ème}$	1.57023e-07	3261.409112	9.74018e+06	21635.111487	4949.878258
	$19^{ème}$	6.90545e-07	4527.274681	1.13803e+07	25567.005027	6143.642470
	$25^{ème}$	1.63378e-06	8610.326003	2.07468e+07	32501.740810	8873.854482
	Moyenne	3.81567e-07	3619.335066	1.05132e+07	21030.406402	5161.075479
	Variance	4.60849e-07	1621.557782	4.37557e+06	6441.561675	1450.382821
FEs=10^5	1^{er}	0.0	0.065053	813066.45845	516.631486	1812.453648
	$7^{ème}$	0.0	0.214957	1.54066e+06	948.127901	2794.833156
	$13^{ème}$	0.0	0.329139	1.90716e+06	1447.308771	3426.708974
	$19^{ème}$	0.0	0.459690	2.27242e+06	2054.595836	4445.213415
	$25^{ème}$	0.0	1.238312	3.76743e+06	4088.917498	5819.294765
	Moyenne	0.0	0.412722	2.01593e+06	1551.897873	3573.855339
	Variance	-	0.288669	698066.45436	804.590438	1001.152670
FEs=$3 \cdot 10^5$	1^{er}	0.0	9.07801e-08	598704.350987	25.455964	1216.232084
	$7^{ème}$	0.0	1.61207e-06	855999.379857	55.108733	2346.496762
	$13^{ème}$	0.0	4.95458e-06	1014045.152060	105.279604	2659.422989
	$19^{ème}$	0.0	9.42962e-06	1247628.169888	155.578752	3298.160834
	$25^{ème}$	0.0	3.0105e-05	1887991.141906	263.511924	4732.928081
	Moyenne	0.0	7.19735e-06	1069031.474667	114.536412	2904.395914
	Variance	-	7.73998e-06	334378.671805	64.802493	893.729250

TABLE B.11 – Erreurs obtenues par TRIBES pour les fonctions multimodales F6 à F12 pour FEs=10^3, 10^4, 10^5 et $3 \cdot 10^5$ en dimension 30.

FEs		F6	F7	F8	F9	F10	F11	F12
FEs=10^3	1^{er}	3.92325e+06	40.013150	21.056059	187.265639	285.218958	31.896924	108258.095532
	$7^{ème}$	2.0412e+07	71.737478	21.171188	249.604655	306.145836	35.902835	157464.201553
	$13^{ème}$	3.50137e+07	85.845944	21.221481	271.948330	325.744326	38.507720	191766.228493
	$19^{ème}$	6.24164e+07	118.854991	21.242706	287.091759	340.021783	41.047662	247644.941821
	$25^{ème}$	2.03865e+08	160.406095	21.298876	406.081078	438.236754	45.843919	451068.981703
	Moyenne	4.99783e+07	96.136898	21.204342	273.795732	331.165661	38.871644	215511.037627
	Variance	4.22058e+07	33.263711	0.059272	43.602115	34.758146	3.900798	81816.336469
FEs=10^4	1^{er}	17.014731	0.085397	21.032120	42.936466	80.173776	17.007092	3611.325757
	$7^{ème}$	189.176584	0.221678	21.058639	61.621876	112.631522	23.086002	14344.047438
	$13^{ème}$	416.425645	0.392264	21.115257	76.893785	128.923325	27.936516	27461.024901
	$19^{ème}$	1350.193152	0.663108	21.156435	92.878949	182.153706	28.776337	57245.400760
	$25^{ème}$	15045.306282	0.991563	21.181138	162.958231	246.566326	30.678365	143722.805979
	Moyenne	2659.469329	0.438602	21.109398	81.094363	148.598773	25.920437	37867.177531
	Variance	4236.307159	0.260645	0.047673	26.525312	51.585323	3.984833	32126.004765
FEs=10^5	1^{er}	14.509209	1.21747e-09	20.839995	42.566535	35.555125	14.705651	591.258305
	$7^{ème}$	22.218218	2.80406e-09	20.961109	49.747867	52.991327	18.163141	4521.349844
	$13^{ème}$	23.409837	1.81492e-08	21.018591	59.861496	70.524538	20.389028	7906.540961
	$19^{ème}$	108.194483	0.00776076	21.045619	68.652044	78.348288	23.403583	15837.006895
	$25^{ème}$	260.359554	0.0392017	21.066397	86.147111	123.191450	26.988445	38892.189697
	Moyenne	71.144465	0.00633127	20.989665	60.061898	68.788406	20.671717	10975.585704
	Variance	71.878558	0.0108424	0.066620	12.161238	17.885204	3.358757	9347.985658
FEs=$3 \cdot 10^5$	1^{er}	11.151063	1.13687e-13	20.815151	36.617389	32.471423	13.148180	423.138620
	$7^{ème}$	17.913102	1.42109e-13	20.907250	45.768041	46.330144	17.221012	1441.010871
	$13^{ème}$	19.051047	2.27374e-13	20.962332	54.309993	50.842266	18.760161	2401.051302
	$19^{ème}$	28.042458	0.00739604	20.990927	62.349902	62.444743	20.308537	7512.223871
	$25^{ème}$	162.687090	0.0392017	21.054735	77.728616	72.461759	24.281864	27042.396094
	Moyenne	42.957510	0.00462371	20.940385	54.915138	53.290049	18.886322	6125.519784
	Variance	45.734870	0.00853559	0.065990	10.860816	12.316491	2.767368	7332.809051

TABLE B.12 – Erreurs obtenues par TRIBES pour les fonctions unimodales F1 à F5 pour FEs=10^3, 10^4, 10^5 et $5 \cdot 10^5$ en dimension 50.

FEs		F1	F2	F3	F4	F5
FEs=10^3	1^{er}	7373.798751	66512.243201	1.608408e+08	1.096512e+05	19776.750982
	$7^{ème}$	13582.060330	123132.752712	3.490499e+08	1.541903e+05	24905.440048
	$13^{ème}$	14654.659667	132489.995334	3.995339e+08	1.775259e+05	26795.888081
	$19^{ème}$	18078.306341	154687.092015	5.436261e+08	1.992400e+05	30405.058354
	$25^{ème}$	23029.561123	177734.220606	8.970324e+08	2.453915e+05	39602.474627
	Moyenne	15373.138183	136021.355994	4.470123e+08	1.785500e+05	28010.523749
	Variance	3553.869792	24836.430592	1.731368e+08	3.708685e+04	5055.179819
FEs=10^4	1^{er}	0.00045	18343.212051	2.073663e+07	5.316309e+04	8895.099403
	$7^{ème}$	0.002793	26256.755860	3.725160e+07	7.464497e+04	10867.563346
	$13^{ème}$	0.012681	33668.457975	4.771993e+07	8.701098e+04	11796.904609
	$19^{ème}$	0.038681	38466.704353	5.798409e+07	9.582370e+04	13032.312161
	$25^{ème}$	0.095290	43600.143258	9.830602e+07	1.173356e+05	20798.492757
	Moyenne	0.022326	32404.253572	5.054105e+07	8.593429e+04	12528.211571
	Variance	0.024181	7381.423033	1.763190e+07	1.562124e+04	2725.159823
FEs=10^5	1^{er}	0.0	546.036631	7.639610e+06	2.303105e+04	5990.765568
	$7^{ème}$	0.0	942.168600	9.828907e+06	2.861164e+04	6663.180346
	$13^{ème}$	0.0	1113.865921	1.117745e+07	3.834594e+04	7506.657817
	$19^{ème}$	0.0	1450.544282	1.313159e+07	4.049924e+04	9513.361510
	$25^{ème}$	0.0	2334.117640	1.801836e+07	4.818047e+04	11020.584031
	Moyenne	0.0	1256.383381	1.168230e+07	3.554161e+04	8136.961363
	Variance	-	511.503268	2.705968e+06	7.935071e+03	1559.783989
FEs=$5 \cdot 10^5$	1^{er}	0.0	0.444382	7.639610e+06	3.929082e+03	5260.547389
	$7^{ème}$	0.0	1.011717	9.828907e+06	8.238943e+03	6139.722171
	$13^{ème}$	0.0	1.319648	1.117745e+07	9.877399e+03	6741.550803
	$19^{ème}$	0.0	1.880692	1.313159e+07	1.207335e+04	8347.882926
	$25^{ème}$	0.0	2.925234	1.801836e+07	1.588878e+04	9986.382681
	Moyenne	0.0	1.527317	1.168230e+07	1.003315e+04	7168.083978
	Variance	-	0.773144	2.705968e+06	2.710695e+03	1342.863057

TABLE B.13 – Erreurs obtenues par TRIBES pour les fonctions multimodales F1 à F5 pour FEs=10^3, 10^4, 10^5 et $5 \cdot 10^5$ en dimension 50.

FEs		F6	F7	F8	F9	F10	F11	F12
FEs=10^3	1er	6.676345e+08	456.524380	21.234798	465.537510	513.461528	65.753077	1.678901e+06
	7ème	1.502741e+09	557.430595	21.311694	553.900449	625.415037	71.808955	1.945224e+06
	13ème	2.102731e+09	647.983077	21.346268	595.439355	660.080508	76.316146	2.300503e+06
	19ème	3.203172e+09	836.367432	21.377617	643.693590	687.156062	79.698272	2.569254e+06
	25ème	7.085885e+09	1222.275318	21.407934	879.884348	767.535534	85.009848	3.441769e+06
	Moyenne	2.567254e+09	688.664200	21.338811	596.145274	658.790550	75.932692	2.363485e+06
	Variance	1.566098e+09	193.401528	0.043641	82.419408	52.848078	5.342803	4.933425e+05
FEs=10^4	1er	6.102731e+01	0.816631	21.109296	148.236247	148.236247	34.243716	1.040067e+05
	7ème	7.367784e+02	0.972842	21.228355	376.164593	376.164593	49.407092	2.168960e+05
	13ème	1.516126e+03	1.094367	21.264091	453.379153	453.379153	52.859016	3.310868e+05
	19ème	5.897766e+03	1.143562	21.286178	487.744376	487.744376	57.316236	3.871801e+05
	25ème	1.710479e+04	1.460009	21.327707	530.327958	530.327958	68.796546	7.210531e+05
	Moyenne	4.447202e+03	1.079263	21.251033	413.006171	413.006171	53.309136	3.278379e+05
	Variance	5.420401e+03	0.139955	0.048455	102.862691	102.862691	8.805442	1.442562e+05
FEs=10^5	1er	41.431073	2.907517e-06	20.405250	86.563856	79.786767	31.436250	2.748311e+04
	7ème	44.562834	1.161803e-04	21.152316	99.614556	120.221918	38.677564	9.110961e+04
	13ème	45.558088	7.462253e-03	21.181590	123.076870	129.999699	40.724507	1.464058e+05
	19ème	100.111610	9.925737e-03	21.208740	145.046664	147.253598	43.990335	2.512531e+05
	25ème	933.549769	1.479109e-02	21.253827	164.427758	212.920642	49.109166	6.820534e+05
	Moyenne	135.990197	6.369441e-03	21.145569	123.107381	137.292408	40.875080	1.890965e+05
	Variance	198.555711	5.068086e-03	0.157824	24.416071	28.455030	4.355602	1.355189e+05
FEs=$5 \cdot 10^5$	1er	36.409515	3.410605e-13	20.262322	81.842922	55.686656	31.436208	1.790812e+04
	7ème	38.880127	4.831691e-13	21.093476	95.353745	80.591477	35.548696	5.492813e+04
	13ème	39.625544	7.396040e-03	21.113697	112.144946	87.556288	39.387344	1.330062e+05
	19ème	41.959751	9.857285e-03	21.159861	124.494016	102.935947	40.414953	1.526228e+05
	25ème	213.409772	1.477978e-02	21.184178	148.818196	136.309043	47.766014	3.377268e+05
	Moyenne	57.607742	5.323914e-03	21.084618	110.142961	91.540077	38.483802	1.351737e+05
	Variance	44.178837	4.961702e-03	0.172698	19.008936	17.366791	3.760477	8.656101e+04

TABLE B.14 – Erreurs moyennes en dimension 30.

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12
BLX-GL50	9.87e-9	9.83e-9	3.11e+3	16.84	332.56	2.59e-7	9.07e-9	20.94	15.10	35.19	24.74	9.52e+3
BLX-MA	9.98e-9	8.71e-6	8.77e+5	39.66	2.18e+3	49.54	0.013	20.71	0.68	90.58	31.13	4.39e+3
CoEVO	0.79	0.44	367.13	4.79e+3	8.34e+3	1.21e+3	0.14	20.90	1.31e+2	232.35	37.70	1.01e+5
DE	0.0	5.49e-8	2.89e+5	0.504	2.35e+2	3.77	0.96	20.9	0.0	61.60	32.60	8.43e+3
DMS-L-PSO	-	-	-	-	-	-	-	-	-	-	-	-
EDA	48.83	161.25	3.75e+6	1.28e+3	6.19e+3	1.82e+5	26.63	20.94	229.79	238.62	39.76	4.47e+5
G-CMA-ES	5.42e-9	6.22e-9	5.55e-9	1.11e+4	8.62e-9	5.90e-9	5.31e-9	20.10	0.93	1.65	5.48	4.43e+4
K-PCX	9.47e-9	9.85e-9	57.9	1.11e+3	2.04e+3	1.75	1.50e-2	20.00	0.27	0.517	29.50	1.68e+3
L-CMA-ES	5.28e-9	6.93e-9	5.18e-9	9.26e+7	8.30e-9	6.31e-9	6.48e-9	20.00	2.91e+2	5.63e+2	15.20	1.32e+4
L-SaDE	-	-	-	-	-	-	-	-	-	-	-	-
SPC-PNX	9.34e-9	6.94e-7	1.10e+6	8.13e-7	4.23e+3	619.78	0.0159	20.93	23.93	60.297	18.09	1.31e+4
SPSO	0.0	0.0	4.21e+5	0.71	3.92e+3	0.81	9.79e-3	20.92	43.40	96.06	34.32	8.14e+4
TRIBES	0.0	7.19e-6	1.06e+6	114.53	2.90e+3	42.95	4.60e-3	20.94	54.91	53.29	18.88	6.12e+3

TABLE B.15 – Erreurs moyennes en dimension 50.

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12
BLX-GL50	-	-	-	-	-	-	-	-	-	-	-	-
BLX-MA	-	-	-	-	-	-	-	-	-	-	-	-
CoEVO	-	-	-	-	-	-	-	-	-	-	-	-
DE	-	-	-	-	-	-	-	-	-	-	-	-
DMS-L-PSO	-	-	-	-	-	-	-	-	-	-	-	-
EDA	-	-	-	-	-	-	-	-	-	-	-	-
G-CMA-ES	5.87e-9	7.86e-9	6.14e-9	4.68e+5	2.85	7.13e-9	7.22e-9	20.1	1.39	1.72	11.7	2.27e+5
K-PCX	-	-	-	-	-	-	-	-	-	-	-	-
L-CMA-ES	6.20e-9	7.96e-9	6.04e-9	4.46e+8	3.27	7.12e-9	7.49e-9	20.0	5.67e+2	1.48e+3	34.1	8.93e+4
L-SaDE	-	-	-	-	-	-	-	-	-	-	-	-
SPC-PNX	-	-	-	-	-	-	-	-	-	-	-	-
SPSO	0.0	1.91e-8	1.18e+6	3.04e+3	8.22e+3	4.31	9.85e-3	21.11	129.00	106.78	58.76	5.45e+5
TRIBES	0.0	1.52	1.16e+7	1.00e+4	7.16e+3	57.60	5.32e-3	21.08	110.14	91.54	38.48	1.35e+5

Références bibliographiques

[Adenso-Diaz et al., 2006] B. Adenso-Diaz, M. Laguna. (2006). "Fine-tuning of algorithms using fractional experimental design and local search", Operations Researchs, Vol. 54, N°1, 2006.

[Alba, 2005] E. Alba. (2005). "Parallel Metaheuristics : A New Class of Algorithms", 2005, John Wiley & sons.

[Andreani et al., 2001] P. Andreani, H. Sjoland. (2001). "Noise Optimisation of an Inductively Degenerated CMOS Low Noise Amplifier", IEEE Transactions on circuits and systems, Vol. 48, pp. 835-841, 2001.

[Angeline, 1998] P. Angeline. (1998). "Evolutionary optimization versus particle swarm optimization", Proceedings of the Evolutionary Programming VII, pp. 601-610, 1998, Springer.

[Auger et al., 2005] A. Auger, S. Kern, N. Hansen. (2005). "A Restart CMA Evolution Strategy with Increasing Population Size", Proceedings of 2005 Congress on Evolutionary Computation, 2005, pp. 1769-1776, IEEE Computer Society.

[Auger et al., 2005b] A. Auger, S. Kern, N. Hansen. (2005). "Performance Evaluation of an Advanced Local Search Evolutionary Algorithm", Proceedings of 2005 Congress on Evolutionary Computation, 2005, pp. 1777-1784, IEEE Computer Society.

[Avriel, 2003] M. Avriel. (2003). "Nonlinear Programming : Analysis and Methods", 2003, Dover Publishing.

[Azencott, 1992] R. Azencott. (1992). "Simulated Annealing : Parallelization Techniques", 1992, John Wiley & sons.

[Ballester et al., 2005] P. J. Ballester, J. Stephenson, J. N. Carter, K. Gallagher. (2005). "Real-Parameter Optimization Performance Study on the CEC'2005 benchmark with SPC-PNX", Proceedings of 2005 Congress on Evolutionary Computation, 2005, pp. 498-505, IEEE Computer Society.

[Banks et al., 2007] A. Banks, J. Vincent, C. Anyakoha. (2007). "A review of Particle Swarm Optimization. Part I: background and development", Natural Computing, Vol. 6, N°4, pp. 467-484, 2007.

[Banks et al., 2007b] A. Banks, J. Vincent, C. Anyakoha. (2007). "A review of Particle Swarm Optimization. Part II: hybridisation, combinatorial, multicriteria and constrained optimization, and indicative applications", Natural Computing, Vol. 7, N°1, pp. 109-124, 2007.

[Bartz-Beielstein et al., 2003] T. Bartz-Beielstein, P. Limbourg, K.E. Parsopoulos, M.N. Vrahatis, J. Mehnen, K. Schmitt. (2003). "Particle swarm optimizers for Pareto optimization with enhanced archiving techniques", Proceedings of the 2003 Congress on Evolutionary Computation, Vol. 3, pp. 1780-1787, IEEE Press.

[Battiti, 1996] R. Battiti. (1996). "Reactive search: toward self tuning heuristics", Modern Heuristic Search Methods, pp. 61-83, John Wiley & Sons.

[Baumgartner et al., 2004] U. Baumgartner, C. Magele, W. Renhart. (2004). "Pareto optimality and particle swarm optimization", IEEE Transactions on Magnetics, Vol. 40, N°2, pp. 1172-1175, 2004.

[Bavelas, 1950] A. Bavelas. (1950). "Communication patterns in task-oriented groups", Journal of the Acoustical Society of America, Vol. 22, pp. 271-282, 1950.

[Bazi et al., 2007] Y. Bazi, L. Bruzzone, F. Melgani. (2007). "Image thresholding based on the EM algorithm and the generalized Gaussian distribution", Pattern Recognition Journal, Vol. 40, pp. 619–634, 2007.

[Benlian et al., 2007] X. Benlian, W. Zhiquan. (2007). "A multiobjective ACO-based data association method for bearings-only multi-target tracking", Communications in Nonlinear Science and Numerical Simulation, Vol. 12, pp. 1360-1369, 2007.

[Ben Salem et al., 2006] S. Ben Salem, M. Fakhfakh, (et al.). (2006). "A High Performances CMOS CCII and High Frequency Applications", Journal of Analog Integrated Circuits and Signal Processing, Vol. 49, N° 1, 2006.

[Berro, 2001] A. Berro. (2001). "Optimisation multiobjectif et stratégies d'évolution en environnement dynamique", Thèse de doctorat de l'Université des Sciences Sociales de Toulouse I, 2001.

[Bertsekas, 2000] D.P. Bertsekas. (2000). "Dynamic Programming and Optimal Control", Vols. 1 & 2, 2nd ed, 2000, Athena Scientific.

[Beyer, 2001] H.G. Beyer. (2001). "The theory of evolution strategies", Natural Computing Series, 2001, Springer.

[Biggs et al., 1976] N.L. Biggs, E.K. Lloyd, R.J. Wilson. (1976). "Graph Theory 1736-1936", 1976, Clarendon Press.

[Birattari et al., 2002] M. Birattari, T. Stützle, L. Paquete, K. Varrentrapp. (2002). "A racing algorithm for configuring metaheuristics", Proceedings of the Genetic and Evolutionary Computation Conference GECCO 2002, pp. 11-18, Morgan Kaufmann.

[Blum, 2005] C. Blum. (2005). "Ant colony optimization : Introduction and recent trends", Physics of Life Review, Vol. 2, pp. 353-373, 2005.

[Bougahriou et al., 2008] M. Bougharíou, M. Fakhfakh, M. Ben Amor, M. Loulou, Y. Cooren, P. Siarry. (2008). "Optimizing a low noise amplifier for UMTS standard through a heuristic", European Conference on Circuits and Systems for Communications 2008, IEEE Press.

[Branke, 2001] J. Branke. (2001). "Evolutionary Optimization in Dynamic Environments", Genetic Algorirhms and Evolutionary Computation, Vol. 3, 2001, Kluwer Academic Publishers.

[Campana et al., 2006] E.F. Campana, G. Fasano, D. Peri, A. Pinto. (2006). "Particle Swarm Optimization : Efficient Globally Convergent Modifications", Proceedings of the 3rd European Conference on Computational Mechanics, Solids and Coupled Problems in Engineering, 2006, Springer.

[Cerny, 1985] V. Cerny. (1985). "Thermodynamical approach to the travelling salesman problem", Journal of Optimization Theory and Applications, Vol. 45, N° 1, pp. 41-51, 1985.

[Chen et al., 2004] L. Chen, X.H. Xu, Y.X. Chen. (2004). "An Adaptive Ant Colony Clustering Algorithm", Proceedings of the 3rd Conference on Machine Learning and Cybernetics, 2004, pp. 1387-1392, IEEE Press.

[Cheriet et al., 1998] M. Cheriet, J.N. Said, C.Y. Suen. (1998). "A recursive thresholding technique for image segmentation", IEEE Trans. on Image Processing, Vol. 7, N°6, pp. 918-921, 1998.

[Chow et al., 2004] C. Chow, H. Tsui. (2004). "Autonomous agents response learning by multi-species particle swarm optimization", Proceedings of the IEEE Congress on Evolutionary Computation, Vol. 1, pp. 778-785, 2004, IEEE Press.

[Clerc et al., 2002] M. Clerc, J. Kennedy. (2002). "The particle swarm: explosion, stability, and convergence in multi-dimensional complex space", IEEE Transactions on Evolutionary Computation, Vol. 6, pp. 58-73, 2002.

[Clerc, 2003] M. Clerc. (2003). "TRIBES - Un exemple d'optimisation par essaim particulaire sans paramère de contrôle", Conférence OEP'03, 2 Octobre , 2003, Paris, France.

[Clerc, 2005] M. Clerc. (2005). "Binary Particle Swarm Optimisers: toolbox, derivations, and mathematical insights", `https://hal.archives-ouvertes.fr/hal-00122809`.

[Clerc, 2006] M. Clerc. (2006). "Particle Swarm Optimization", International Scientific and Technical Encyclopaedia, 2006, John Wiley & sons.

[Cocosco et al., 2003] C.A. Cocosco, A.P. Zijdenbos, A.C. Evans. (2003). "A fully automatic and robust brain MRI Tissue Classification", Medical Image Analysis, Vol. 7, pp. 513–527, 2003.

[Coello Coello et al., 2002] C.A. Coello Coello, D.A. Veldhuizen, G.B. Lamont. (2002). "Evolutionary Algorithms for Solving Multi-Objective Problems", Kluwer Academic Publishers.

[Coello Coello et al., 2002b] C.A. Coello Coello, M. Salazar Lechuga. (2002). "MOPSO : a proposal for multiple objective particle swarm optimization", Proceedings of the 2002 IEEE Congress on Evolutionary Computation, Vol. 2, pp. 1051-1056, IEEE Press.

[Coello Coello et al., 2004] C.A. Coello Coello, G. Toscano Pulido, M. Salazar Lechuga. (2004). "Handling multiple objectives with particle swarm optimization", IEEE Transactions on Evolutionary Computation, Vol. 8, N° 3, pp. 256-279, 2004.

[Collette et al., 2002] Y. Collette, P. Siarry. (2002). "Optimisation multiobjectif", 2002, Eyrolles.

[Colorni et al., 1992] A. Colorni, M. Dorigo, V. Maniezzo. (1992). "Distributed Optimization by Ant Colonies", Proceedings of the 1$^{\text{rst}}$ European Conference on Artificial Life, 1992, pp. 134-142, Elsevier Publishing.

[Conn et al., 1996] A. R. Conn et al.. (1996). "Optimization of custom MOS circuits by transistor sizing", Proceedings of the International Conference on Computer-Aided Design ICCAD96, pp. 174-190, 1996, IEEE Press.

[Cooren et al., 2007] Y. Cooren, M. Clerc, P. Siarry. (2007). "Initialization and Displacements of the Particles in TRIBES, a Parameter-free Particle Swarm Optimization Algorithm", Adaptive and Multilevel Metaheuristics, pp. 199-219, Springer.

[Cooren et al., 2007b] Y. Cooren, M. Fakhfakh, M. Loulou, P. Siarry. (2007). "Optimizing second generation current conveyors using particle swarm optimization", Proceedings of the 19th IEEE International Conference on Microelectronics, pp. 365-368, IEEE Press.

[Cooren et al., 2008] Y. Cooren, M. Clerc, P. Siarry. (2008). "Performance Evaluation of TRIBES, an Adaptive Particle Swarm Optimization Algorithm", Swarm Intelligence, Springer. *En révision.*

[Cooren et al., 2008b] Y. Cooren, M. Clerc, P. Siarry. (2008). "MO-TRIBES, an adaptive multiobjective particle swarm optimization algorithm", Computational Optimization and Applications, Springer. *Soumis.*

[Cooren et al., 2008c] Y. Cooren, A. Nakib, P. Siarry. (2008). "Image Thresholding using TRIBES, a parameter-free Particle Swarm Optimization Algorithm", Proceedings of the International Conference on Learning and Intelligent Optimization (LION II), pp. 81-94, LNCS 5313, Springer.

[Courat et al., 1994] J. Courat, G. Raynaud, I. Mrad, P. Siarry. (1994). "Electronic component model minimisation based on Log Simulated Annealing", IEEE Transactions on Circuits and Systems, Vol. 41, N° 12, pp. 790-795, 1994.

[Deb, 1999] K. Deb. (1999). "Multi-Objective Genetic Algorithms: Problem Difficulties and Construction of Test Problems", Evolutionary Computation, Vol.7, N°3, pp. 205-230, 1999.

[Deb et al., 2002] K. Deb, S. Agrawal, A. Pratap, T. Meyarivan. (2002). "A Fast Elitist Non-Dominated Sorting Genetic Algorithm for Multiobjective Optimization : NSGA-II", IEEE Transactions on Evolutionary Computation, Vol. 5, N°3, pp. 115-148, 2002.

[Di Caro et al., 1998] G. Di Caro, M. Dorigo. (1998). "Ant colonies for adaptive routing in packet-switched communications networks", Proceedings of the 5$^{\text{th}}$ International Conference on Parallel Problem Solving from Nature, 1998, LNCS 1498, pp. 673-682, Springer.

[Di Caro et al., 1998b] G. Di Caro, M. Dorigo. (1998). "AntNet: Distributed Stigmergetic Control for Communications Networks", Journal of Artificial Intelligence Research (JAIR), Vol. 9, pp. 317-365, 1998.

[Doerner et al., 2004] K. Doerner et al.. (2004). "Pareto Ant Colony Optimization : a metaheuristic approach to multiobjective project portofolio selection", Annals of Operations Research, Vol. 131, pp. 79-99, 2004.

[Doerner et al., 2006] K. Doerner et al.. (2006). "Pareto Ant Colony Optimization with ILP preprocessing in multiobjective project portofolio selection", European Journal of Operational Research, Vol. 171, pp. 830-841, 2006.

[Dorigo et al., 1996] M. Dorigo, V. Maniezzo, A. Colorni. (1996). "Ant System : optimization by a colony of cooperating agents", IEEE Transactions on Man. Cyber., Part B, Vol. 26, N°1, pp. 29-41, 1996.

[Dorigo et al., 2005] M. Dorigo, C. Blum. (2005). "Ant colony optimization theory : A survey", Theoretical Computer Science, Vol. 344, pp. 243-278, 2005.

[Drapaca et al., 2005] C.S. Drapaca, V. Cardenas, C. Studholme. (2005). "Segmentation of tissue boundary evolution from brain MR image sequences using multi-phase level sets", Computer Vision and Image Understanding, Vol. 100, pp. 312–329, 2005.

[Eberhart et al., 1995] R.C. Eberhart, J. Kennedy. (1995). "A new optimizer using particle swarm theory", Proceedings of the 6$^{\text{th}}$ International Symposium on Micro Machine and Human Science, 1995, pp. 39-43, IEEE Press.

[Eberhart et al., 1996] R.C. Eberhart, P. Simpson, R. Dobbins. (1996). "Computational PC Tools", chapter 6, pp. 212-226, 1996, AP Professional.

[Eberhart et al., 2000] R.C. Eberhart, Y. Shi. (2000). "Comparing inertia weights and constriction factors in particle swarm optimization", Proceedings of the 6$^{\text{th}}$ IEEE Congress on Evolutionary Computation, pp. 84-88, 2000, IEEE Press.

[Eiben et al., 2003] A.E. Eiben, J.E. Smith. (2003). "Introduction to Evolutionary Computing", Natural Computing Series, 2003, Springer.

[Engrand, 1997] P. Engrand. (1997). "A multi-objective optimization approach based on simulated annealing and its application to nuclear fuel management", Proceedings of the 5$^{\text{th}}$ International Conference on Nuclear Engineering, pp. 416-423, 1997, American Society Of Mechanical Engineers.

[Fan et al., 2001] H.Y. Fan, Y. Shi. (2001). "Study on V_{max} of particle swarm optimization", Proceedings of the 2001 Workshop on Particle Swarm Optimization, Indiana University-Purdue University Indianapolis Press.

[Fogel et al., 1966] L.J. Fogel, A.J. Owens, M.J. Walsh. (1966). "Artificial Intelligence through Simulated Evolution", 1966, John Wiley & sons.

[Fonseca et al., 1995] C.M. Fonseca, P.J. Flemming. (1995). "An overview of evolutionary algorithms in multiobjective optimization", Evolutionary Computation, Vol. 3, N°1, pp. 205-209, 1995.

[Fonseca et al., 1996] C.M. Fonseca, P.J. Flemming. "On the performance assessment and comparison of stochastic multiobjective optimizers", Proceedings of the Parallel Problem Solving from Nature IV Conference, pp. 584-593, 1996, Lecture Notes in Computer Science, Springer.

[Förster et al., 2007] M. Förster, B. Bickel, B. Hardung, G. Kókai. (2007). "Self-adaptive ant colony optimization applied to function allocation in vehicle networks", Proceedings of the 9th annual Conference on Genetic and Evolutionary Computation, 2007, pp. 1991-1998, ACM Press.

[Fraser, 1957] A.S. Fraser. (1957). "Simulation of genetic systems by automatic digital computers", Australian Journal of Biological Sciences, Vol. 10, pp. 484-491, 1957.

[Friedman, 1937] M. Friedman. (1937). "The use of ranks to avoid the assumption of normality implicit in the analysis of variance", Journal of the American Statistical Association, Vol. 32 (200), pp. 675–701, 1937.

[Friesz et al., 1993] T.L. Friesz et al.. (1993). "The multiobjective equilibrium network design problem revisited : a simulated annealing approach", European Journal of Operational Research, Vol. 65, pp. 44-57, 1993.

[Gagné et al., 2004] C. Gagné, M. Gravel, W.L. Price. (2004). "Optimisation multiobjectif à l'aide d'un algorithme de colonies de fourmis", Inf. Syst. and Operational Research, Vol. 42, N°1, pp. 23-42, 2004.

[García-Martínez et al., 2005] C. García-Martínez, M. Lozano. (2005). "Hybrid Real-Coded Genetic Algorithms with Female and Male Differentiation", Proceedings of 2005 Congress on Evolutionary Computation, 2005, pp. 896-903, IEEE Computer Society.

[Gielen et al., 1991] G. Gielen, W. Sansen. (1991). "Symbolic Analysis for Automated Design of Analog Integrated Circuits", 1991, Kluwer Academic Publishers.

[Glover, 1986] F. Glover. (1986). "Future paths for integer programming and links to artificial intelligence", Computers and Operations Research, Vol. 13, pp. 533-549, 1986.

[Glover et al., 1997] F. Glover, M. Laguna. (1997). "Tabu Search", 1997, Kluwer Academic Publishers.

[Goldberg et al., 1987] D.E. Goldberg, J. Richardson. (1987). "Genetic Algorithms whih Sharing for Multimodal Function Optimization", Proceedings of the 2nd International Conference on Genetic Algorithms, 1987, pp. 41-49, Lawrence Earlbaum Associates.

[Goldberg, 1989] D.E. Goldberg. (1989). "Genetic Algorithms in Search, Optimization and Machine Learning", 1989, Addison-Wesley.

[Gonzalez et al., 2002] R.C. Gonzalez, R.E. Woods. (2002). "Digital image processing", 2002, Prentice Hall.

[Goss et al., 1989] S. Goss, S. Aron, J.L. Deneubourg, J.M. Pasteels. (1989). "Self-Organized Shortcuts in the Argentine Ant", Naturwissenchaften, Vol. 76, pp. 579-581, 1989.

[Graeb et al., 2001] H. Graeb, S. Zizala, J. Eckmueller, K. Antreich. (2001). "The sizing rules method for analog integrated circuit design", IEEE/ACM International Conference on Computer-Aided Design, ICCAD'01, pp. 343-349, 2001, IEEE Press.

[Harary, 1994] F. Harary. (2004). "Graph Theory", 1994, Addison-Wesley, Reading.

[Hastings, 1970] W.K. Hastings. (1970). "Monte Carlo sampling method using Markov chains and their applications", Biometrica, Vol. 57, 1970.

[Holland, 1973] J.H. Holland. (1973). "Genetic Algorithms and the optimal allocation of trials", SIAM Journal of Computing, Vol. 2, pp. 88-105, 1973.

[Holland, 1992] J.H. Holland. (1992). "Adaptation in Natural and Artificial Sytems", 2nd edition, 1992, MIT Press.

[Hu et al., 2002] X. Hu, R.C. Eberhart. (2002). "Multiobjective optimization using dynamic neighborhood particle swarm optimization", Proceedings of the 2002 IEEE Congress on Evolutionary Computation, Vol. 2, pp. 1677-1681, IEEE Press.

[Hu et al., 2003] X. Hu, R.C. Eberhart, Y. Shi. (2003). "Particle Swarm with Extended Memory for Multiobjective Optimization", Proceedings of the 2003 IEEE Swarm Intelligence Symposium, pp. 193-197, IEEE Press.

[Hung et al., 1990] K. K. Hung, P. K. Ko, C. Hu, Y. C. Cheng. (1990). "A physics-based MOSFET noise model for circuit simulators", IEEE Trans. Electronic Devices, Vol. 37, pp. 1323–1333, 1990.

[Hutter et al., 2006] F. Hutter, Y. Hamadi, H.H. Hoos, K. Leyton-Brown. (2006). "Performance Prediction and Automated Tuning of Randomized and Parametric Algorithms", Proceedings of the Principles and Practice of Constraint Programming Conference, 2006, pp. 213-228, LNCS 4204, Springer.

[Ingber, 1996] L. Ingber. (1996). "Adaptive Simulated Annealing (ASA): lessons learned", Control and Cybernetics, Vol. 25, N° 1, pp. 33-54, 1996.

[Iqbal et al., 2006] M. Iqbal, M.A. Montes de Oca. (2006). "An estimation of distribution particle swarm optimization algorithm", Proceedings of the 5th International Workshop on Ant Colony Optimization and Swarm Intelligence, pp. 72-83, 2006, LNCS 4150, Springer.

[Ishibushi et al., 1998] H. Ishibushi, T. Murata. (1998). "A multiobjective genetic local search algorithm and its application to flowshop scheduling", IEEE Transactions on Systems, Man and Cybernetics, Vol. 28, N°3, pp. 1732-1743, 1998.

[Janson et al., 2005] S. Janson, M. Middenhorf. (2005). "A hierarchical particle swarm optimizer and its adaptive variant", Proceedings of the 2004 Evoworkshops, pp. 513-524, 2005, LNCS 3005, Springer.

[Jin et al., 2001] Y. Jin, T. Okabe, B. Sendhoff. (2001). "Dynamic weighted aggregation for evolutionary multiobjective optimization : why does it work and how ?", Proceedings of the Genetic and Evolutionary Conference, pp. 1042-1049, Morgan Kaufman Publishers.

[Kamber et al., 2000] M. Kamber, R. Shinghal, D.L. Collins, G.S. Francis, A.C. Evans. (2000). "Model-based segmentation of multiple sclerosis lesions in magnetic resonance brain images", IEEE Trans. on Med. Imaging, Vol. 14, pp. 442–453, 2000.

[Kapur et al., 1985] J.N. Kapur, P.K. Sahoo, A.C.K. Wong. (1985). "A new method for gray-level picture thresholding using the entropy of the histogram", Computer Vision, Graphics and Image Processing, Vol. 29, pp. 273-285, 1985.

[Kennedy et al., 1995] J. Kennedy, R.C. Eberhart. (1995). "Particle Swarm Optimisation", Proceedings of the IEEE International Conference On Neural Networks, 1995, pp. 1942-1948, IEEE Press.

[Kennedy, 1998] J. Kennedy. (1998). "The behavior of particles", Proceedings of the 7th Conference on Evolutionary Computation, pp. 581-589, LNCS, Springer.

[Kennedy, 1999] J. Kennedy. (1999). "Small worlds and mega-minds : effects of neighborhood topology on particle swarm performance", Proceedings of the 1999 IEEE Congress on Evolutionary Computation, pp. 1931-1938, 1999, IEEE Press.

[Kennedy et al., 2001] J. Kennedy, R.C. Eberhart, Y. Shi. (2001). "Swarm Intelligence", 2001, Morgan Kaufmann Academic Press.

[Kennedy, 2003] J. Kennedy. (2003). "Bare bones particle swarms", Proceedings of the 2003 IEEE Swarm Intelligence Symposium, pp. 80-87, 2003, IEEE Press.

[Kirkpatrick et al., 1983] S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi. (1983). "Optimization by simulated annealing", Science, Vol. 220, N° 4598, pp. 671-680, 1983.

[Kursawe, 1990] F. Kursawe. (1990). "A variant of evolution strategies for vector optimization", Proceedings of Parallel Problem Solving for Nature Conference, pp. 193-197, 1990, Lecture Notes in Computer Science 496, Springer.

[Larrañaga et al., 2001] P. Larrañaga, J.A. Lozano. (2001). "Estimation of Distributions Algorithms, a new tool for evolutionary computation", 2001, Kluwer Academic Publishers.

[Laumanns et al., 2001] M. Laumanns, K. Deb, L. Thiele, E. Zitzler. (2001). "Scalable test problems for Evolutionary Multi-Objective Optimization", Technical Report 112, Institut für Technische Informatik und Kommunikationsnetze, ETH Zürich, Ch-8092, Zürich, July, 2001.

[Li, 2003] X. Li. (2003). "A non-dominated sorting particle swarm optimizer for multiobjective optimization", Proceedings of the 2003 Genetic and Evolutionary Conference, pp. 37-48, LNCS 2723, Springer.

[Liang et al., 2005] J.J. Liang, P.N. Suganthan. (2005). "Dynamic multiswarm particle swarm optimizer", Proceedings of the IEEE Swarm Intelligence Symposium, pp. 1772-1779, 2005, IEEE Press.

[Liang et al., 2005b] J.J. Liang, P.N. Suganthan, K. Deb. (2005). "Novel Composition Test Functions for Numerical Global Optimization", Proceedings of the 2005 Swarm Intelligence Symposium, 2005, pp. 68-75, IEEE Press.

[Liang et al., 2005c] J.J. Liang, P.N. Suganthan. (2005). "Dynamic Multi-Swarm Particle Swarm Optimizer with Local Search", Proceedings of 2005 Congress on Evolutionary Computation, 2005, pp. 522-528, IEEE Computer Society.

[Loulou et al., 2002] M. Loulou, S. Ait Ali, M. Fakhfakh, N. Masmoudi. (2002). "An optimized methodology to design CMOS operational amplifier", IEEE International Conference on Microelectronic, ICM 2002, pp. 14-17, 2002, IEEE Press.

[Maron et al., 1994] O. Maron, A.W. Moore. (1994). "Hoeffding races : Accelerating model selection search for classification and function approximation", Advances in Neural Information Processing Systems, Vol. 6, pp. 59–66, 1994, Morgan Kaufmann Publishers.

[Medeiro et al., 1994] E. Medeiro et al., (1994). "Global design of analog cells using statistical optimization techniques", Analog Integrated Circuits and Signal Processing, Vol. 6, pp. 179-195, 1994, Kluwer Academic Publishers.

[Metropolis et al., 1953] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, E. Teller. (1953). "Equation of state calculations by fast computing machines", Journal of Chemical Physics, Vol. 21, pp. 1087-1092, 1953.

[Miranda et al., 2002] V. Miranda, N. Fonseca. (2002). "New evolutionary particle swarm algorithm applied to voltage/VAR control", Proceedings of the 14^{th} Power Systems Computation Conference, 2002, pp. 1-6, Sevilla, Spain.

[Molina et al., 2005] D. Molina, F. Herrera, M. Lozano. (2005). "Adaptive Local Search Parameters for Real-Coded Memetic Algorithms", Proceedings of 2005 Congress on Evolutionary Computation, 2005, pp. 888-895, IEEE Computer Society.

[Mülhenbein et al., 1996] H. Mülhenbein, G. Paaβ. (1996). "From recombination of genes to the estimation of distributions I. Binary parameters", Proceedings of the International Conference on Parallel Problem Solving from Nature IV, 1996, LNCS 1411, pp. 178-187, Springer.

[Murata et al., 2002] Y. Murata and al. (2002). "Agent Oriented Self Adaptive Genetic Algorithm", Proceedings of the IASTED Communications and Computer Networks, 2002, pp. 348-353, Acta Press.

[Murgasova et al., 2006] M. Murgasova, L. Dyet, D. Edwards, M. Rutherford, J.V. Hajnal, D. Rueckert. (2006). "Segmentation of Brain MRI in Young Children", Proceedings of the 9th Int. Conf. MICCAI Copenhagen, pp. 687–694, 2006, LNCS, Springer.

[Nair, 1992] V.N. Nair. (1992). "Taguchi's parameter design : a panel discussion", Technometrics, Vol. 34, pp. 127-161, 1992.

[Nakib et al., 2007] A. Nakib, H. Oulhadj, P. Siarry. (2007). "Microscopic image segmentation based on two dimensional exponential entropy with hybrid microcanonical annealing", Proceedings of Int. Conf. IAPR-MVA 2007, pp. 420–423, 2007.

[Nakib et al., 2007b] A. Nakib, Y. Cooren, H. Oulhadj, P. Siarry. (2007). "Magnetic Resonance Image Segmentation based on Two-dimensional Exponential Entropy", Proceedings of the International Conference on Artificial Evolution, pp. 50-61, Springer.

[Nelder et al., 1965] J.A. Nelder, R. Mead. (1965). "A simplex method for function minimization", Computer Journal, Vol. 7, pp. 308-313, 1965.

[Ng, 2006] H. Ng. (2006). "Automatic thresholding for defect detection", Pattern Recognition Letters, Vol. 27, pp. 1644–1649, 2006.

[Niu et al., 2005] B. Niu, Y. Zhu, X. He, W. Henry. (2005). "MCPSO : A multi-swarm cooperative particle swarm optimizer", Applied Mathematics and Computation, Vol. 185, N° 2, pp. 1050-1062, 2005.

[Nocedal et al., 1999] J. Nocedal, S.J. Wright. (1999). "Numerical Optimization", 1999, Springer.

[Okabe et al., 2003] T. Okabe, Y. Jin, B. Senhoff. (2003)."A critical survey of performances indices for multi-objective optimization", Proceedings of the 2003 IEEE Congress on Evolutionary Computation, pp. 878-885, 2003, IEEE Press.

[Papers, 2005] Papers of CEC'05. (2005).
`http://www3.ntu.edu.sg/home/EPNSugan/`

[Pareto, 1896] V. Pareto. (1896). "Cours d'économie politique", Rouge, 1896.

[Parsopoulos et al., 2002] K.E. Parsopoulos, M.N. Vrahatis. (2002). "Particle Swarm Optimization in Multiobjective Problems", Proceedings of the 2002 ACM Symposium on Applied Computing, pp. 603-607, ACM Press.

[Parsopoulos et al., 2004] K.E. Parsopoulos, D. Tasoulis, M.N. Vrahatis. (2004). "Multiobjective optimization using parallel vector evaluated particle swarm optimization", Proceedings of the IASTED International Conference on Artificial Intelligence and Applications, Vol. 2, pp. 823-828, Acta Press.

[Peer et al., 2003] E.S. Peer, F. Van den Bergh, A.P. Engelbrecht. (2003). "Using neighborhoods with the guaranteed convergence PSO", Proceedings of the IEEE Swarm Intelligence Symposium 2003 (SIS 2003), 2003, pp. 235-242, IEEE Press.

[Peng-Yeng, 2007] Y. Peng-Yeng. (2007). "Multilevel minimum cross entropy threshold selection based on particle swarm optimization", Applied Mathematics and Computation, Vol. 184, pp. 503–513, 2007.

[Peram et al., 2003] T. Peram, K. Veeramachaneni, C. Mohan. (2003). "Fitness-distance ratio based particle swarm optimization", Proceedings of the IEEE Swarm Intelligence Symposium, pp. 174-181, 2003, IEEE Press.

[Poli et al., 2004] R. Poli, C.R. Stephens. (2004). "Constrained molecular dynamics as a search and optimization tool", Proceedings of the 7th European Conference on Genetic Programming, pp. 150-161, 2004, Springer.

[Posik, 2005] P. Posik. (2005). "Real Parameter Optimization Using Mutation Step Co-evolution", Proceedings of 2005 Congress on Evolutionary Computation2005, pp. 872-879, IEEE Computer Society.

[Price, 1996] K.V. Price. (1996). "Differential evolution : a fast and simple numerical optimizer", 1996 Biennial Conference of the North American Fuzzy Information Processing Society, pp. 524-527, 1996, IEEE Press.

[PSC, 2006] Particle Swarm Central. (2006). `http://www.particleswarm.info/Standard_PSO_2006.c`

[Otsu, 1979] N.A. Otsu. (1979). "A Threshold Selection Method from Gray Level Histograms", IEEE Trans. on Syst., Man and Cyb., Vol. 9, N°1, pp. 62-66, 1979.

[Ourique et al., 2002] C. Ourique, E.J. Biscaia, J. Pinto. (2002). "The use of particle swarm optimization for dynamical analysis in chemical processes", Computers & Chemical Engineering, Vol. 26, N°12, pp. 1783-1793, 2002.

[Ozcan et al., 1999] E. Ozcan, C.K. Mohan. (1999). "Particle Swarm Optimization : surfing the waves", Proceeding of the 1999 IEEE Congress on Evolutionary Computation, pp. 1939-1944, IEEE Press.

[Qiaio et al., 2007] Y. Qiao, Q. Hu, G. Qian, S. Luo, W.L. Nowinski. (2007). "Thresholding based on variance and intensity contrast", Pattern Recognition, Vol. 40, pp. 596–608, 2007.

[Qin et al., 2005] A. K. Qin, P. N. Suganthan. (2005). "Self-adaptive Differential Evolution Algorithm for Numerical Optimization", Proceedings of 2005 Congress on Evolutionary Computation, 2005, pp. 1785-1791, IEEE Computer Society.

[Raquel et al., 2005] C.R. Raquel, J.P.C. Naval. (2005). "An effective use of crowding distance in multiobjective particle swarm optimization", Proceedings of the 2005 Genetic and Evolutionary Computation Conference, pp. 257-264, ACM Press.

[Ray et al., 2002] T. Ray, K.M. Liew. (2002). "A swarm metaphor for multiobjective design optimization", Engineering Optimization, Vol. 34, N°2, pp. 141-153, 2002.

[Razavi, 1998] R. Razavi. (1998). "RF Microelectronics", 1998, Prentice Hall Press.

[Rechenberg, 1965] I. Rechenberg. (1965). "Cybernetic Solution Path of an Experimental Problem", 1965, Royal Aircarft Establishment Library Translation.

[Reyes-Sierra et al., 2006] M. Reyes-Sierra, A. Coello Coello. (2006). "Multi-Objective Particle Swarm Optimizers : A survey of the state-of-the-art", International Journal of Computational Intelligence Research, Vol. 2, N°3, pp. 287-308, 2006.

[Reynolds, 1987] C.W. Reynolds. (1987). "Flocks, herds and schools: a distributed behavioral model", Computer Graphics, Vol. 21, N°4, pp.25-34, 1987.

[Richardson et al., 1989] J.T. Richardson et al.. (1989). "Some guidelines for genetic algorithms with penalty functions", Proceedings of the 3rd International Conference on Genetic Algorithms, pp. 191-197, 1989, Morgan Kaufmann Publishers.

[Robinson et al., 2002] J. Robinson, J. Sinton, Y. Rahamt-Samii. (2002). "Particle swarm, genetic algorithm, and their hybrids : optimization of a profiled corrugated horn antenna", Proceedings of the IEEE International Symposium on Antennas and Propogation, pp. 314-317, 2002, IEEE Press.

[Rönkkönen et al., 2005] J. Rönkkönen, S. Kukkonen, K. V. Price. (2005). "Real-Parameter Optimization with Differential Evolution", Proceedings of 2005 Congress on Evolutionary Computation, 2005, pp. 506-513, IEEE Computer Society.

[Russell et al., 2003] S.J. Russell, P. Norvig. (2003). "Artificial Intelligence: A Modern Approach", 2nd ed., pp. 111-114, Prentice Hall.

[Sahoo et al., 1998] P.K. Sahoo et al.. (1998). "A survey of thresholding techniques", Computer Vision, Graphics, and Image Processing, Vol. 41, pp. 233-260, 1998.

[Sahoo et al., 2006] K.P. Sahoo, G. Arora. (2006). "Image thresholding using two dimensional Tsallis-Havrda-Charvát entropy", Pattern Recognition Letters, Vol. 27, pp. 520–528, 2006.

[Sawai et al., 1999] H. Sawai, S. Adachi. (1999). "Genetic Algorithm Inspired by Gene Duplication", Proceedings of the 1999 Congress on Evolutionary Computation, 1999, pp. 480-487, IEEE Computer Society.

[Schaffer, 1985] J.D. Schaffer. (1985). "Multiple objective optimization with vector evaluated genetic algorithm", Proceedings of the 1rst International Conference on Genetic Algorithm, pp. 93-100, Lawrence Erlbaum Associates.

[Schmid, 2000] H.Schmid. (2000). "Approximating the Universal Active Element", IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal Processing, Vol. 47, N° 11, 2000.

[Schnecke et al., 1996] V. Schnecke, O. Vornberger. (1996). "An Adaptive Parallel Genetic Algorithm for VLSI- Layout Optimization", Proceedings of the 4th International Conference on Parallel Problem Solving from Nature, 1996, pp. 859-868, LNCS, Springer.

[Schrijver, 1998] A. Schrijver. (1998). "Theory of Linear and Integer Programming", 1998, John Wiley & sons.

[Seevinck, 2000] E. Seevinck. (2000). "CMOS Translinear Circuits for Minimum Supply Voltage", IEEE Transactions on Circuits and Systems-II : Analog and Digital Signal Processing, Vol. 47, N°12, pp. 1560–1564, 2000.

[Serafini, 1992] P. Serafini. (1992). "Simulated annealing for multiple objective optimization problems", Proceedings of the 10th International Conference on Multiple Criteria Decision, pp. 87-96, Springer.

[Serra et al., 1997] P. Serra, A.F. Stanton, S. Kais. (1997). "Method for global optimization", Physical Review, Vol. 55, pp. 1162-1165, 1997.

[Sezgin et al., 2004] M. Sezgin, B. Sankur. (2004). "Survey over image thresholding techniques and quantitative performance evaluation", Journal of Electronic Imaging, Vol. 13, pp. 146–165, 2004.

[Shelokar et al., 2004] P.S. Shelokar, P. Siarry, V.K. Jayaraman, B.D. Kulkarni. (2004). "Particle swarm and ant colony algorithms hybridized for improved continuous optimization", Applied Mathematics and Computation, Volume 188, N° 1, Pages 129-142, 2004.

[Shi et al., 1998] Y. Shi, R. Eberhart. (1998). "Parameter Selection in Particle Swarm Optimization", Proceedings of the 7th Annual Conference on Evolutionary Programming, 1998, pp. 591-600, LNCS 1447, Springer.

[Shi et al., 2001] Y. Shi, R.C. Eberhart. (2001). "Fuzzy Adaptive Particle Swarm Optimization", Proceedings of 2001 Congress on Evolutionary Computation, 2001, pp. 101-106, IEEE Computer Society.

[Siarry, 1994] P. Siarry. (1994). "La méthode du recuit simulé en électronique. Adaptation et accélération. Comparaison avec d'autres méthodes d'optimisation. Applications dans d'autres domaines", Habilitation à diriger des recherches en sciences physiques, Université de Paris Sud, 1994.

[Siarry et al., 2003] P. Siarry, J. Dréo, A. Pétrowski, E. Taillard. (2006). "Métaheuristiques pour l'optimisation difficile", 2003, Eyrolles.

[Silveira et al., 1996] F. Silveira, D. Flandre, P. G. A. Jespers. (1996). "A gm/Id based methodology for the design of CMOS Analog Circuits and its Application to the Synthesis of a SOI micropower OTA", IEEE Journal of Solid State Circuits, Vol. 31, N°9, 1996.

[Sinha et al., 2005] A. Sinha, S. Tiwari, K. Deb. (2005). "A Population-Based, Steady-State Procedure for Real-Parameter Optimization", Proceedings of 2005 Congress on Evolutionary Computation, 2005, pp. 514-521, IEEE Computer Society.

[Song et al., 2006] Z. Song, N. Tustison, B. Avants, J.C. Gee. (2006). "Integrated Graph Cuts for Brain MRI Segmentation", Proceedings of the 9th Int. Conf. MICCAI, Copenhagen, Denmark, pp. 831–838, 2006.

[Srinivasan et al., 2003] D. Srinivasan, T.H. Seow. (2003). "Particle swarm inspired evolutionary algorithm for multiobjective optimization problem", Proceedings of the 2003 Congress on Evolutionary Computation, Vol. 3, pp. 2292-2297, IEEE Press.

[Srivians et al., 1995] N. Srivinas, K. Deb. (1995). "Multiobjective Optimization using Non-Dominated Sorting in Genetic Algorithms", Evolutionary Computation, Vol. 2, N°8, pp. 221-248, 1995.

[Suganthan, 1999] P.N. Suganthan. (1999). "Particle Swarm Optimizer with neighborhood operator", Proceedings of the 1999 IEEE Conference on Evolutionary Computation, pp. 1958-1962, IEEE Press.

[Suganthan et al., 2005] P.N. Suganthan and al. (2005). "Problem Definitions and Evaluation Criteria for the CEC'2005 Special Session on Real-Parameter Optimization", Technical Report, Nanyang Technological University, Singapore, May 2005, AND KanGAL Report #2005005, IIT Kanpur, India.
`http://www.ntu.edu.sg/home/epnsugan/index_files/CEC-05/Tech-Report-May-30-05.pdf`

[Synder et al., 1990] W. Synder, G. Bilbro. "Optimal thresholding : A new approach", Pattern Recognition Letters, Vol. 11, pp. 803-810, 1990.

[Talbi, 1999] E.G. Talbi. (1999). "Métaheuristiques pour l'optimisation combinatoire multiobjectif : Etat de l'art", CNET, PE :98-757.33, 1999.

[Talbi, 2002] E.G. Talbi. (2002). "A taxonomy of Hybrid Metaheuristics", Journal of Heuristics, Vol. 8, N°5, pp. 541-564, 2002.

[Tfaili, 2007] W. Tfaili. (2007). "Conception d'un algorithme de colonie de fourmis pour l'optimisation continue dynamique", Thèse de doctorat de l'Université de Paris 12-Val de Marne, 2007.

[Toumazou et al., 1993] C. Toumazou, F.J.Lidgey, D.G.Haigh. (1993). "Analog Integrated Circuits : The current mode approach", IEEE circuit and systems series 2 (books) 1993, IEEE Press.

[Trelea, 2003] I.C. Trelea. (2003). "The particle swarm optimization algorithm: convergence analysis and parameter selection", Information Processing Letters, Vol. 85, pp. 317-325, 2003.

[Triki et al., 2004] E. Triki, Y. Collette, P. Siarry. (2005). "A theoretical study on the behavior of simulated annealing leading to a new cooling schedule", European Journal of Operational Research, Vol. 166, N° 1, 2005.

[Ulungu et al., 1999] E.L. Ulungu, J. Teghem, P. Fortemps, D. Tuyttens. (1999). "MOSA method: a tool for solving multiobjective combinatorial optimization problems", Journal of Multicriteria Decision Analysis, Vol. 20, pp. 221-236, 1999.

[Van den Bergh, 2002] F. Van den Bergh. (2002). "An analysis of Particle Swarm Optimizers", PhD thesis, University of Pretoria, South Africa, 2002.

[Warfield et al., 2000] S.K. Warfield, M. Kaus, F.A. Jolesz, R. Kikinis.(2000). "Adaptive template moderate spatially varying statistical classification", Medical Image analysis, Vol. 4, pp. 43–55, 2000.

[Watts et al., 1998] D.J. Watts, S.H. Strogatz. (1998). "Collective dynamics of "small worlds" networks", Nature, Vol. 393, pp. 440-442, 1998.

[Wilson, 1975] E.O. Wilson. (1975). "Sociobiology: The new synthesis", 1975, Belknap Press.

[Yasuda et al., 2004] K. Yasuda, N. Iwasaki. (2004). "Adaptive particle swarm optimization using velocity information of swarm", Proceedings of the IEEE Conference on System, Man and Cybernetics, 2004, pp. 3475-3481, IEEE Press.

[Ye et al., 2002] X.F. Ye, W.J. Zhang, Z.L. Yang. (2002). "Adaptive Particle Swarm Optimization on Individual Level", Proceedings of the International Conference on Signal Processing (ICSP), 2002, pp. 1215-1218, IEEE Press.

[Yuan et al., 2005] B. Yuan, M. Gallagher. (2005). "Experimental Results for the Special Session on Real-Parameter Optimization at CEC'2005: A Simple, Continuous EDA", Proceedings of 2005 Congress on Evolutionary Computation, 2005, pp. 1792-1799, IEEE Computer Society.

[Zitzler et al., 1999] E. Zitzler, L. Thiele. (1999). "Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach", IEEE Transactions on Evolutionary Computation, Vol. 3, N°4, pp. 257-271, 1999.

[Zitzler et al., 2000] E. Zitzler, K. Deb, L. Thiele. (2000). "Comparison of Multiobjective Evolutionary Algorithms : Empirical Results", Evolutionary Computation, Vol. 8, N°2, pp. 173-195, 2000.

[Zitzler et al., 2002] E. Zitzler, M. Laumanns, L. Thiele. (2002). "SPEA2 : Improving the Strengh Pareto Evolutionary Algorithm for Multiobjective Optimization", Proceedings of the Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems Conference, pp. 95-100, CIMNE, Barcelona, Spain.

[Zhang et al., 2003] W.J. Zhang, X.F. Xie. (2003). "DEPSO: hybrid particle swarm with differential evolution operator", Proceedings of the IEEE Conference on Systems, Man and Cybernetics, pp. 3816-3821, 2003, IEEE Press.

[Zhang et al., 2003b] W. Zhang, Y. Liu, M. Clerc. (2003). "An adaptive PSO algorithm for real power optimization", Proceedings of the APSCOM (Advances in Power System Control Operation and Management) Conference, S6: Application of Artificial Intelligence Technique (part I), 2003, pp. 302-307, IEEE Press.

[Zheng et al., 2003] Y. Zheng, L. Ma, L. Zhang, J. Qian. (2003). "On the convergence analysis and parameter selection in particle swarm optimization", Proceedings of International Conference on Machine Learning and Cybernetics, 2003, pp. 1802-1807, IEEE Press.

[Zhong et al., 2004] W.C. Zhong, J. Liu, Z. Xue, L.C. Jiao. (2004). "A multiagent genetic algorithm for global numerical optimization", IEEE Transactions on Systems, Man and Cybernetics, Vol. 34, pp. 1128-1141, 2004.