Perfectionnement de métaheuristiques pour l'optimisation continue

Ilhem Boussaid

UNIVERSITE PARIS-EST CRÉTEIL

ÉCOLE DOCTORALE (ED 532)

MATHÉMATIQUES ET SCIENCES ET TECHNOLOGIES DE L'INFORMATION ET DE LA COMMUNICATION (MSTIC)

THÈSE DE DOCTORAT

AVEC L'UNIVERSITÉ DES SCIENCES ET DE LA TÉCHNOLOGIE HOUARI BOUMEDIENE

SPÉCIALITÉ INFORMATIQUE

intitulé

PERFECTIONNEMENT DE MÉTAHEURISTIQUES POUR L'OPTIMISATION CONTINUE

soutenue publiquement le 29 juin 2013 par

Ilhem BOUSSAÏD

devant le jury composé de

M. Mohammed SLIMANE	Professeur des universités	Univ. Tours	**Rapporteur**
M. El-Ghazali TALBI	Professeur des universités	Univ. Lille	**Rapporteur**
Mme Aïcha MOKHTARI	Professeur des universités	USTHB	**Examinateur**
Mme Dalila BOUGHACI	Maître de Conférences	USTHB	**Examinateur**
M. Mohamed AHMED-NACER	Professeur des universités	USTHB	**Co-directeur de thèse**
M. Patrick SIARRY	Professeur des universités	Univ. Créteil	**Directeur de thèse**

Résumé

Les métaheuristiques sont des algorithmes génériques, souvent inspirés de la nature, conçues pour résoudre des problèmes d'optimisation complexes. Parmi les métaheuristiques les plus récentes, nous retenons celle basée sur la théorie de la biogéographie insulaire : *Biogeography-based optimization* (BBO).

Dans cette thèse, nous considérons à la fois les problèmes d'optimisation globale à variables continues avec et sans contraintes. De nouvelles versions hybrides de BBO sont proposées comme des solutions très prometteuses pour résoudre les problèmes considérés. Les méthodes proposées visent à pallier les inconvénients de la convergence lente et du manque de diversité de l'algorithme BBO. Dans la première partie de cette thèse, nous présentons la méthode que nous avons développée, issue d'une hybridation de BBO avec l'évolution différentielle (DE) pour résoudre des problèmes d'optimisation sans contraintes. Nous montrons que les résultats de l'algorithme proposé sont plus précis, notamment pour des problèmes multimodaux, qui sont parmi les problèmes les plus difficiles pour de nombreux algorithmes d'optimisation. Pour résoudre des problèmes d'optimisation sous contraintes, nous proposons trois nouvelles variantes de BBO. Des expérimentations ont été menées pour rendre compte de l'utilité des méthodes proposées.

Dans une deuxième partie, nous nous intéressons à l'étude des capacités des méthodes proposées à résoudre des problèmes d'optimisation, issus du monde réel. Nous nous proposons d'abord de résoudre le problème d'allocation optimale de puissance pour la détection décentralisée d'un signal déterministe dans un réseau de capteurs sans fil, compte tenu des fortes contraintes en ressources énergétiques et en bande passante des nœuds répartis. L'objectif est de minimiser la puissance totale allouée aux capteurs, tout en gardant la probabilité d'erreur de détection au dessous d'un seuil requis. Dans un deuxième temps, nous nous focalisons sur la segmentation d'images en niveaux de gris par seuillage multi-niveaux. Les seuils sont déterminés de manière à maximiser l'entropie floue. Ce problème d'optimisation est résolu en appliquant une variante de BBO (DBBO-Fuzzy) que nous avons développée. Nous montrons l'efficacité de la méthode proposée aux travers de résultats expérimentaux.

Mots clés : *optimisation difficile, optimisation continue, métaheuristiques, algorithme d'optimisation basé sur la biogéographie (BBO), algorithme à évolution différentielle (DE), allocation optimale de puissance, réseaux de capteurs sans fil, segmentation, seuillage, entropie floue.*

Abstract

Metaheuristics are general algorithmic frameworks, often nature-inspired, designed to solve complex optimization problems. Among representative metaheuristics, Biogeography-based optimization (BBO) has been recently proposed as a viable stochastic optimization algorithm.

In this PhD thesis, both unconstrained and constrained global optimization problems in a continuous space are considered. New hybrid versions of BBO are proposed as promising solvers for the considered problems. The proposed methods aim to overcome the drawbacks of slow convergence and the lack of diversity of the BBO algorithm. In the first part of this thesis, we present the method we developed, based on an hybridization of BBO with the differential evolution (DE) algorithm, to solve unconstrained optimization problems. We show that the results of the proposed algorithm are more accurate, especially for multimodal problems, which are amongst the most difficult-to-handle class of problems for many optimization algorithms. To solve constrained optimization problems, we propose three new variations of BBO. Our extensive experimentations successfully demonstrate the usefulness of all these modifications proposed for the BBO algorithm.

In the second part, we focus on the applications of the proposed algorithms to solve real-world optimization problems. We first address the problem of optimal power scheduling for the decentralized detection of a deterministic signal in a wireless sensor network, with power and bandwidth constrained distributed nodes. The objective is to minimize the total power spent by the whole sensor network while keeping the detection error probability below a required threshold. In a second time, image segmentation of gray-level images is performed by multilevel thresholding. The optimal thresholds for this purpose are found by maximizing the fuzzy entropy. The optimization is conducted by a newly-developed BBO variants (DBBO-Fuzzy). We show the efficiency of the proposed method through experimental results.

Keywords : *Hard optimization, continuous optimization, metaheuristics, biogeography-based optimization (BBO) algorithm, differential evolution (DE) algorithm, optimal power allocation, wireless sensor network (WSN), segmentation, thresholding, fuzzy entropy.*

Remerciements

Je tiens tout d'abord à adresser mes plus vifs remerciements aux directeurs de cette thèse, Patrick Siarry et Mohamed Ahmed-Nacer, pour leur direction avisée et exigeante, leurs qualités humaines et scientifiques et leur soutien constant tout au long de cette thèse.

I am also very grateful to Amitava Chatterjee for his scientific advice and knowledge and many insightful suggestions.

Je tiens à remercier sincèrement Aïcha Mokhtari qui m'a honoré en acceptant d'être la présidente de mon jury de thèse. Je voudrais aussi exprimer ma gratitude à Dalila Boughaci, à Mohammed Slimane et à El-Ghazali Talbi qui ont bien voulu prendre de leur temps pour évaluer mon travail et donner leur avis. Leur jugement m'est très important.

Merci aussi à mes collègues de l'USTHB qui en font un endroit agréable à exercer les activités de chercheur et d'enseignement. Je remercie également mes collègues thésards, pour leur amitié, notamment Abbas El-Dor, Julien Lepagnot, Mustafa Dakkak, Noureddine Belgacem, Wafa Chaabane, Vera Demerjian, Nacer Benaichouche.

Il m'est impossible de ne pas remercier l'ensemble des membres du LiSSi pour le travail dans la bonne humeur et les nombreux prétextes trouvés pour se réunir autour d'un café ou d'un thé à la menthe. Je me dois de remercier Patricia Jamin, Sandrine David, Yasmina, Oléna Tankyevych, Éric Petit, Amir Nakib, Steve Guyot et tous ceux que j'oublie, mais qui ne m'en voudront pas ou si peu.

Un grand « Merci » à mon cher époux, qui a toujours su me redonner confiance dans les instants de doute, pour son enthousiasme contagieux et pour nos moments de complicité et de bonheur qu'il m'offre chaque jour.

Merci à Véronique pour son soutien de qualité, pour Marie-Thé (ma correctrice des fautes d'orthographes) pour sa gentillesse et sa générosité.

Ces remerciements ne peuvent s'achever, sans une pensée pour mes parents, ainsi que mes frères et sœurs qui ont plus particulièrement assuré le soutien affectif de ce travail doctoral.

À Giulia-Fairouz, notre rayon de soleil

Paris, 07 juillet 2013 I. B.

Table des matières

Introduction générale

L'optimisation combinatoire est un outil essentiel pour le scientifique et pour l'ingénieur en particulier. Elle couvre un large éventail de techniques et fait toujours l'objet de recherches intensives. Ses domaines d'application sont extrêmement variés : optimisation d'un trajet, de la forme d'un objet, du contrôle aérien, du choix des investissements économiques, etc. L'optimisation de ces systèmes permet de trouver une configuration idéale, d'obtenir un gain d'effort, de temps, d'argent, d'énergie, ou encore de satisfaction. Pour certains de ces problèmes, la solution optimale ne peut être trouvée par des méthodes exactes, au regard des temps d'exécutions prohibitifs que nécessitent les problèmes de taille réelle. C'est le cas des problèmes dits d'*optimisation difficile*.

Dans la première partie de cette thèse, nous présenterons les métaheuristiques, qui sont des méthodes permettant d'obtenir une valeur approchée de la solution optimale en un temps polynomiale. Leur but est de permettre la résolution d'une large gamme de problèmes différents, sans nécessiter de changements profonds dans l'algorithme. Elles s'inspirent généralement d'analogies avec la physique (recuit simulé), avec la biologie (algorithmes évolutionnaires) ou encore l'éthologie (colonies de fourmis, essaims particulaires). Leurs domaines d'application sont vastes et s'étendent souvent bien au-delà des problèmes pour lesquels elles ont été initialement conçues. Un intérêt particulier est apporté à la méthode d'optimisation basée sur la biogéographie (BBO : *Biogeography-Based Optimization*), qui constitue le sujet principal de ce travail de thèse.

L'étude du comportement de BBO sur un *benchmark* de fonctions de test a permis de mettre en évidence certains défauts, conduisant souvent à des solutions sous optimales. Afin d'y remédier, nous proposons une amélioration basée sur une hybridation avec l'évolution différentielle, considéré comme l'un des algorithmes les plus performants dans la littérature pour résoudre des problèmes d'optimisation continue. Nous proposons, dans un deuxième temps, d'étendre le domaine d'application de BBO aux problèmes d'optimisation sous contraintes.

Les applications étudiées dans la deuxième partie de ce manuscrit, portent sur l'utilisation des algorithmes développés pour résoudre des problèmes dans le domaine des réseaux de capteurs sans fil et du traitement d'images.

Dans le domaine des réseaux de capteurs sans fil, la consommation énergétique des capteurs est souvent une préoccupation majeure, étant donné le peu d'énergie dont ils disposent. Les contraintes qui en résultent expliquent la présence d'un centre de fusion au sein du réseau, disposant de plus de ressources, permettant d'agréger/exploiter les données recueillies par l'ensemble des capteurs. Notre travail se situe dans le contexte de *détection décentralisée*, dans laquelle chaque capteur effectue un traitement préliminaire des données observées et transmet une décision locale au centre de fusion, qui prend la décision finale. Le critère de performance que nous proposons d'utiliser repose sur la formulation Bayésienne qui est basée sur la probabilité d'erreur de détection au centre de fusion. Plus particulièrement, nous nous focalisons sur l'optimisation de l'expression de l'énergie consommée tout en minimisant la probabilité d'erreur au centre de fusion. Une résolution analytique du problème est présentée dans le cas où les observations des capteurs sont indépendantes et identiquement distribuées (i.i.d.). Le modèle de corrélation spatiale rend par contre difficile l'obtention d'une expression analytique du problème d'optimisation, dont la résolution s'obtient par une approche numérique à l'aide d'algorithmes d'optimisation stochastiques.

Nous nous sommes également intéressés à la segmentation d'images qui est sans doute la tâche en analyse d'images qui mobilise le plus d'effort. Elle permet d'isoler, de façon aussi exacte que possible, les différents objets présents sur une image. Dans notre travail, nous nous sommes particulièrement intéressés aux méthodes de seuillage, et plus précisément au *seuillage multi-niveaux* (ou *multithresholding*), qui consiste à répartir les pixels de l'image en plusieurs classes. Toutefois, l'ambiguïté d'appartenance d'un pixel aux objets ou au fond pose souvent problème. Cette imprécision, dont sont souvent entachées les données d'images, trouve ses origines à différents niveaux : phénomènes observés, limites des capteurs et acquisitions des données, bruit, nature des images, etc. Nous avons donc pris le parti de focaliser notre travail sur la théorie des sous-ensembles flous, qui est sans conteste la théorie la plus utilisée dans la littérature pour modéliser l'imprécision. En raisonnant avec la logique floue, l'objectif est de trouver les seuils qui minimisent l'incertitude associée à l'image. Cette incertitude est mesurée par l'entropie floue, qui est une extension de l'entropie probabiliste de Shannon aux évènements flous. Les seuils sont donc déterminés de manière à maximiser l'entropie floue résultant du découpage de l'histogramme de l'image en plusieurs classes.

Cette thèse a été préparée conjointement au sein de l'université des Sciences et de la Technologie Houari Boumediene, dans le *Laboratoire des Systèmes Informatiques* (LSI) et à l'université de Paris-Est Créteil, dans le *Laboratoire Images, Signaux et Systèmes Intelligents* (LiSSi, E.A. 3956). Le laboratoire LiSSi travaille en particulier sur des problèmes de recherche opérationnelle, et notamment sur les méthodes d'optimisation, dans le cadre d'applications en traitement

d'images ou dans le domaine médical. Ce travail a été proposé et encadré par Patrick Siarry, professeur et directeur de l'équipe Traitement de l'Image et du Signal. Il a été co-encadré par Mohamed Ahmed-Nacer, professeur et directeur du laboratoire LSI.

Le plan de ce manuscrit est le suivant :

Le premier chapitre, à caractère introductif et bibliographique, est consacré à un état de l'art des différentes approches couramment employées pour résoudre un problème d'optimisation. Bien évidemment, ce chapitre ne prétend pas à l'exhaustivité mais il espère aborder et présenter les principales métaheuristiques. Il introduit la théorie de la complexité et pose les bases théoriques qui seront utilisées par la suite. Nous ferons en particulier une description des problèmes d'optimisation avec contraintes, ainsi qu'une brève introduction à l'optimisation multiobjectif.

Dans le second chapitre, un premier volet sera consacré à l'étude du comportement de BBO dans l'optique de réaliser des gains de performance. Nous en présenterons une analyse expérimentale, en vue d'examiner l'influence de ses différents paramètres. Nous présenterons ensuite l'algorithme d'optimisation *2-StageBBO-DE*, que nous avons élaboré dans le cadre des problèmes d'optimisation continue sans contraintes. Une étude expérimentale et des comparaisons à différentes méthodes de la littérature, via différents *benchmarks* classiques, sont également présentées.

Dans le troisième chapitre, nous présenterons les différents algorithmes d'optimisation développés pour étendre le domaine d'application de l'algorithme BBO aux problèmes d'optimisation sous contraintes. Comme précédemment, nous décrirons en détail les algorithmes proposés, en présenterons une analyse expérimentale, ainsi qu'une comparaison avec d'autres algorithmes d'optimisation avec contraintes proposés dans la littérature.

Le quatrième chapitre illustre l'utilisation des méthodes développées au chapitre précédent en les appliquant au problème d'allocation optimale de puissance dans un réseau de capteur sans fil. Nous comparerons les résultats obtenus à ceux d'autres méthodes de la littérature, sur les mêmes jeux de données.

Le cinquième et dernier chapitre est consacré à l'application de BBO à la segmentation d'images. Nous feront un nouvel état de l'art, cette fois-ci pour décrire le contexte dans lequel se place l'application. Nous étudierons la façon dont la logique floue peut être utilisée pour la segmentation d'image et nous porterons une attention particulière à l'entropie floue. Les performances de l'approche proposée seront étudiées de façon détaillée.

Nous terminons finalement ce manuscrit par une conclusion dans laquelle nous récapitulerons nos contributions, puis nous proposerons des perspectives, sur la base des travaux effectués.

Chapitre Un

Métaheuristiques d'optimisation : Revue de la littérature

1.1 Introduction

L'optimisation est une discipline en plein essor qui entre en jeu dans beaucoup de domaines, comme dans la conception de circuits électroniques, la recherche opérationnelle, la biologie, mais aussi pour répondre aux besoins croissants des secteurs économique et industriel (maximisation des performances, minimisation des coûts).

Sans prétendre à une totale exhaustivité, ce premier chapitre s'efforce de présenter un état de l'art sur les métaheuristiques pour la résolution de problèmes d'optimisation difficile. On distingue généralement deux grandes familles de métaheuristiques : celles qui manipulent en parallèle toute une population de solutions (on peut citer les algorithmes génétiques, la méthode des colonies de fourmis, l'optimisation par essaim particulaire, etc.) et les autres qui se basent sur l'évolution itérative d'une solution unique (la méthode Tabou et le Recuit Simulé sont des exemples typiques de ces méthodes). Nous invitons le lecteur intéressé par plus de détails à consulter notre article [Boussaïd et al., 2013c].

1.2 Théorie de la complexité

La théorie de la complexité [Papadimitriou, 1994] s'intéresse à l'étude formelle de la difficulté des problèmes en informatique. On se pose alors la question fondamentale de savoir : « *entre différents algorithmes réalisant une même tâche, quel est le plus performant* ? ». Pour comparer les algorithmes entre eux, on pourrait calculer le temps mis par chaque algorithme pour s'exécuter, une fois implémenté sur une machine. Une telle démarche est dépendante des caractéristiques de la machine ou du langage utilisé. Une approche indépendante des facteurs matériels est donc nécessaire. Les mathématiques deviennent alors un outil central dans l'analyse rigoureuse des algorithmes, notamment sous l'impulsion de Donald Knuth, et de son fameux traité « *The Art of Computer Programming* » [Knuth, 1998]. Il s'agit de calculer une mesure de la complexité d'un algorithme, en nombre d'opérations élémentaires nécessaires pour que l'algorithme fournisse la solution du problème à l'utilisateur.

Par ailleurs, certains travaux précurseurs (citons ceux de Stephen Cook [Cook, 1971] et Richard Karp [Karp, 1972]) ont mis en évidence la jonction entre l'optimisation combinatoire et la théorie de la complexité. Ceux-ci montrent en effet que parmi des centaines de problèmes d'optimisation discrète qui étaient l'objet de recherches éparses, il existe une classification fondamentale. On distingue alors la classe P des problèmes qui sont calculables en temps polynomial par une machine de Turing déterministe [1]. Face à la classe P, il y a la très célèbre classe NP (*Non-déterministe Polynomial*) qui est définie à partir d'un modèle de calcul non déterministe. De façon équivalente, c'est la classe des problèmes qui sont calculables par une machine de Turing non déterministe en temps polynomial (c'est-à-dire qu'on peut tester la validité d'une solution du problème en un temps polynomial). La question centrale consiste bien évidemment à savoir si $P = NP$. L'inclusion de P dans NP étant évidente, car un algorithme déterministe est un algorithme non déterministe particulier ce qui, dit en mots plus simples, signifie que si une solution peut être calculée en temps polynomial, alors elle peut être vérifiée en temps polynomial. L'importance majeure de cette question, formellement posée depuis plus de trente ans, n'est plus à démontrer. La conjecture actuelle est plutôt de dire que $P \neq NP$ [2].

Une autre question fondamentale consiste alors à savoir s'il existe un problème de NP plus *difficile* que tous les autres. De tels problèmes seront dits *NP-complets*. On qualifie de *NP-complets* les problèmes décisionnels, c'est-à-dire ceux dont la réponse est de type binaire (*oui/non*, *vrai/faux*, *1/0*, ...). Pour montrer qu'un problème de décision Π est *NP-complet*, il suffit donc de montrer que Π appartient à NP, de choisir un problème Π' connu pour être *NP-complet* et d'établir la relation $\Pi' \prec \Pi$, où $\prec$ dénote la transformation (ou réduction) polynomiale (appelée aussi réduction de Karp [Karp, 1972]). Les problèmes *NP-complets* possèdent deux propriétés intéressantes. Tout d'abord, aucun d'entre eux n'a pu être résolu, à ce jour, par un algorithme polynomial, malgré les efforts persistants de plusieurs générations de chercheurs. Ensuite, si l'on trouvait un algorithme polynomial permettant de résoudre un seul problème *NP-complet*, on pourrait en déduire un autre pour chacun des autres problèmes de cette classe.

On qualifie de *NP-difficiles* les problèmes d'optimisation, c'est-à-dire ceux dont la réponse est de type numérique. À un problème d'optimisation *NP-difficile* on peut associer un problème de décision *NP-complet* dont le but est de déterminer s'il existe une solution pour laquelle la fonction objectif soit supérieure (respectivement inférieure) ou égale à une valeur donnée, mais dire qu'un problème *NP-difficile* est aussi *NP-complet* est un abus de langage. De nombreux

1. Une machine de Turing déterministe a au plus une transition possible, contrairement à une machine de Turing non déterministe, qui peut en avoir plusieurs. Cela signifie qu'alors que les calculs d'une machine de Turing déterministe forment une suite, ceux d'une machine de Turing non déterministe forment un arbre, dans lequel chaque chemin correspond à une suite de calculs possibles.

2. Notons que l'égalité $P = NP$, est un problème ouvert fondamental de l'informatique théorique ; il fait partie des « Problèmes du Millénaire » recensés en sciences mathématiques par la Fondation Clay.

problèmes d'optimisation combinatoire ont été prouvés *NP-difficiles*. Cette difficulté n'est pas seulement théorique, mais se confirme aussi dans la pratique.

1.3 Problème d'optimisation

Un problème d'optimisation se définit comme la recherche, parmi un ensemble de solutions possibles $\mathcal{S}$ (appelé aussi *espace de décision* ou *espace de recherche*), de la (ou des) solution(s) x^* qui rend(ent) minimale (ou maximale) une fonction mesurant la qualité de cette solution. Cette fonction est appelée *fonction objectif* ou *fonction coût*. Si l'on pose $f : \mathcal{S} \longmapsto \mathbb{R}$ la fonction objectif à minimiser (respectivement à maximiser) à valeurs dans $\mathbb{R}$, le problème revient alors à trouver l'optimum $x^* \in \mathcal{S}$ tel que $f(x^*)$ soit minimal (respectivement maximal).

Lorsque l'on veut résoudre un problème d'optimisation, on recherche la meilleure solution possible à ce problème, c'est-à-dire l'*optimum global*. Cependant, il peut exister des solutions intermédiaires, qui sont également des optimums, mais uniquement pour un sous-espace restreint de l'espace de recherche : on parle alors d'*optimums locaux*. Cette notion est illustrée dans la figure 1.1. La seule hypothèse faite sur $\mathcal{S}$ est qu'il s'agit d'un espace topologique, i.e. sur lequel est définie une notion de *voisinage*. Cette hypothèse est nécessaire pour définir la notion de *solutions locales* du problème d'optimisation. On peut alors définir un optimum local (relativement au voisinage V) comme la solution x^* de $\mathcal{S}$ telle que $f(x^*) \leq f(x); \forall x \in V(x^*)$.

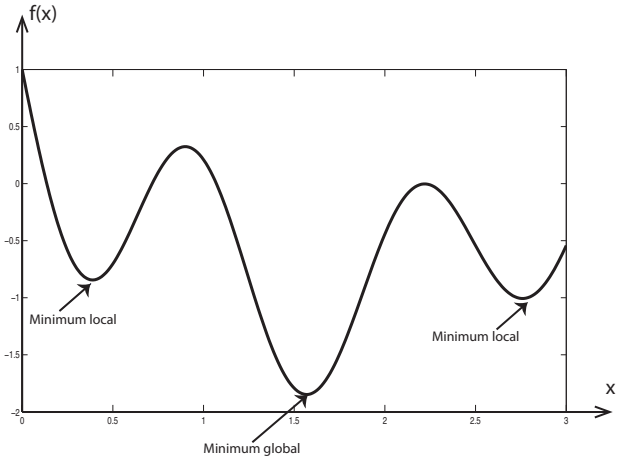


FIGURE 1.1: Différence entre un optimum global et des optima locaux

1.4 Classification des méthodes d'optimisation

La résolution d'un problème d'optimisation est réalisée à l'aide des méthodes d'optimisation dont la classification est illustrée dans la figure 1.2. On distingue en premier lieu l'optimisation continue de l'optimisation discrète (ou combinatoire). Cette première distinction concerne

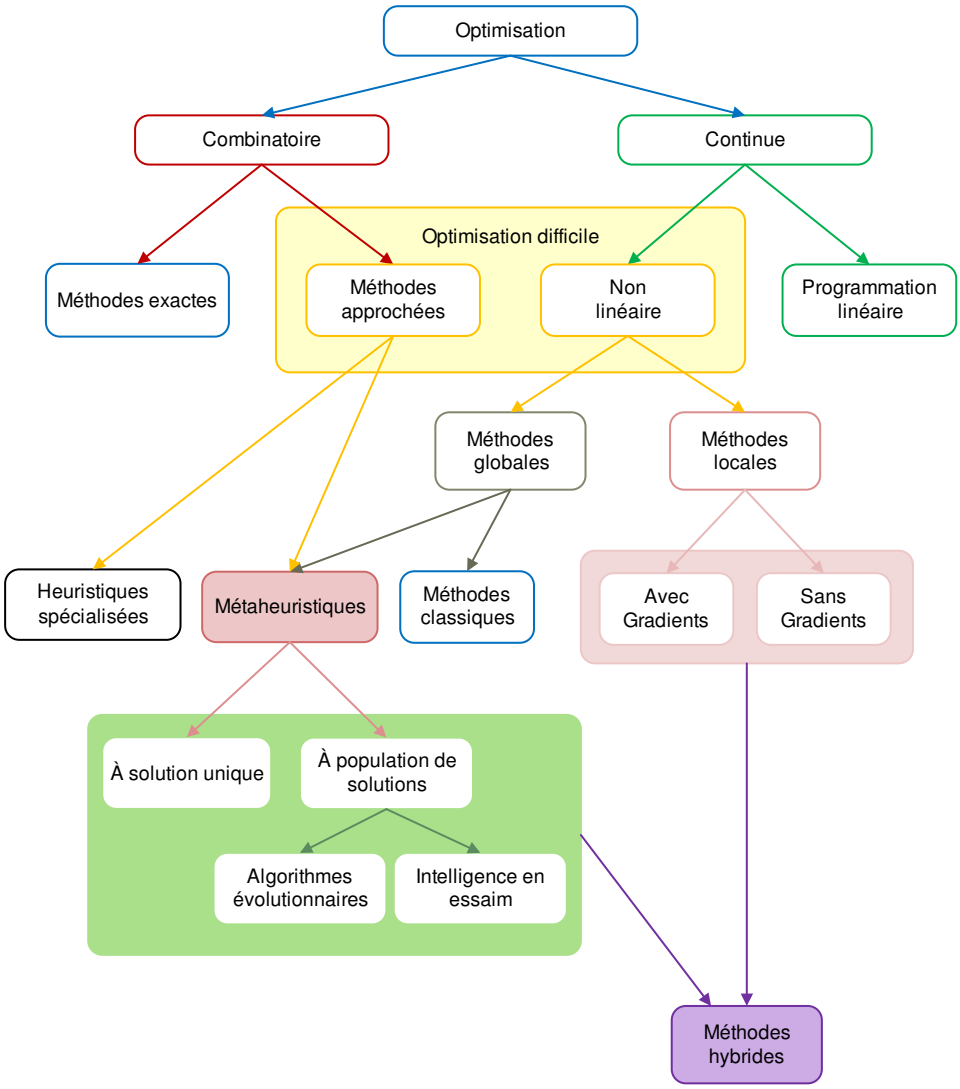


FIGURE 1.2: Classification des méthodes d'optimisation [Dréo et al., 2003]

la nature des « espaces » dans lesquels les variables de décision prennent leurs valeurs : c'est la dichotomie « discret-continu » bien marquée en Mathématiques et qui conditionne évidemment beaucoup les possibilités de recourir à certaines méthodes. Pour l'optimisation continue, on sépare sommairement le cas linéaire (qui relève notamment de la programmation linéaire) du cas non linéaire, où l'on retrouve le cadre de l'optimisation difficile. Pour les problèmes d'optimisation combinatoire de taille raisonnable, les méthodes *exactes* peuvent trouver des solutions optimales. Ces méthodes explorent de façon systématique l'espace des combinaisons jusqu'à trouver une solution optimale. Or, le plus souvent, la combinatoire du nombre des valeurs prises par l'ensemble des variables est « explosive », ce qui exclut l'énumération exhaustive comme méthode de recherche de la solution. Afin de (tenter de) contenir l'explosion combinatoire, ces approches structurent l'espace des combinaisons en arbre et utilisent des techniques d'élagage, pour réduire cet espace, et des heuristiques, pour déterminer l'ordre dans lequel il est

exploré. Cependant, les techniques de filtrage et les heuristiques d'ordre ne réduisent pas toujours suffisamment la combinatoire, et certaines instances de problèmes ne peuvent être résolues en un temps acceptable par ces approches exhaustives. Parmi les méthodes exactes, on peut citer les méthodes de séparation et évaluation, dites méthodes de *Branch and Bound* [Land & Doig, 1960], et la programmation dynamique.

Lorsque l'on dispose d'un temps de calcul limité ou lorsqu'on est confronté à des problèmes difficiles ou de taille importante, on peut avoir recours aux méthodes approchées, en se contentant de rechercher une solution de « bonne qualité » ; dans ce cas le choix est parfois possible entre une *heuristique spécialisée*, entièrement dédiée au problème considéré, et une *métaheuristique* [Dréo et al., 2003].

1.4.1 Heuristiques et Métaheuristiques

On a vu dans la section 1.2 qu'on ne connaît pas d'algorithmes polynomiaux pour les problèmes *NP-difficiles*, et la conjecture $P \neq NP$ fait qu'il est peu probable qu'il en existe. Or, de nombreux problèmes d'optimisation combinatoire sont *NP-difficiles* et ne pourront donc pas être résolus de manière exacte dans un temps *raisonnable*.

Le mot heuristique vient du grec eurisko $\epsilon\upsilon\rho\iota\sigma\kappa\omega$ qui signifie « je trouve » d'où le célèbre Eureka d'Archimède. Une heuristique, ou méthode approximative, est un algorithme qui fournit rapidement (en temps polynomial) une solution réalisable, pas nécessairement optimale, pour un problème d'optimisation *NP-difficile*. On oppose les méthodes approchées aux méthodes exactes, qui trouvent toujours l'optimum si on leur en laisse le temps (énumération complète, méthodes arborescentes, programmation dynamique). Les approches *heuristiques*, contournent le problème de l'explosion combinatoire en faisant délibérément des impasses et n'explorent qu'une partie de l'espace des combinaisons. Une méthode heuristique est généralement conçue pour un problème particulier, en s'appuyant sur sa structure propre. On parle de *métaheuristique* pour les méthodes approximatives générales, pouvant s'appliquer à différents problèmes.

Le mot *métaheuristique* est dérivé de la composition de deux mots grecs : méta, du grec $\mu\epsilon\tau\alpha$ signifiant « au-delà » (ou « à un plus haut niveau ») et heuristique. En effet, ces algorithmes se veulent des méthodes génériques pouvant optimiser une large gamme de problèmes différents, sans nécessiter de changements profonds dans l'algorithme employé. Les métaheuristiques sont en général *non-déterministes*[3], elles peuvent ne pas trouver la solution optimale, et encore moins prouver l'optimalité de la solution trouvée. On peut distinguer les métaheuristiques qui font

3. Un algorithme est dit *non déterministe* lorsque, appliqué sur une instance donnée du problème, donne des résultats qui peuvent varier d'une exécution à l'autre où dont le temps d'exécution varie, contrairement aux algorithmes *déterministes*, qui se comportent toujours de la même façon et produisent toujours le même résultat (heuristique gloutonne).

évoluer une seule solution sur l'espace de recherche à chaque itération et les métaheuristiques à base de population de solutions. En général, les métaheuristiques à base de solution unique sont plutôt axées sur l'*exploitation* de l'espace de recherche, on n'est donc jamais sûr d'obtenir l'*optimum optimorum*. Les métaheuristiques à base de population sont plutôt *exploratoires* et permettent une meilleure diversification de l'espace de recherche.

1.5 Les métaheuristiques à solution unique

Dans cette section, nous présentons les métaheuristiques à base de solution unique, aussi appelées *méthodes de trajectoire*. Contrairement aux métaheuristiques à base de population, les métaheuristiques à solution unique commencent avec une seule solution initiale et s'en éloignent progressivement, en construisant une trajectoire dans l'espace de recherche. Les méthodes de trajectoire englobent essentiellement la méthode de descente, la méthode du recuit simulé, la recherche tabou, la méthode GRASP, la recherche à voisinage variable, la recherche locale itérée, et leurs variantes.

1.5.1 Méthode de descente

La méthode de descente (DM : *Descent method*) est l'une des méthodes les plus simples de la littérature. Elle est également appelée *hill climbing* dans les problèmes de maximisation. Son principe consiste, à partir d'une solution initiale, à choisir à chaque itération un point dans le voisinage de la solution courante qui améliore strictement la fonction objectif. Il existe plusieurs moyens de choisir ce voisin : soit par le choix aléatoire d'un voisin parmi ceux qui améliorent la solution courante (*first improvement*) ; soit en choisissant le meilleur voisin qui améliore la solution courante (*best improvement*). Dans tous les cas, le critère d'arrêt est atteint lorsque plus aucune solution voisine n'améliore la solution courante [Papadimitriou & Steiglitz, 1982].

Le principal inconvénient de la DM est qu'elle reste piégée dans le premier optimum local rencontré. Les méthodes de ce type ne présentent aucune forme de diversification. Une amélioration de cet algorithme consiste à redémarrer plusieurs fois, lorsqu'un optimum local est trouvé, à partir d'une nouvelle solution générée aléatoirement. On parle alors d'algorithme de descente avec relance (*multiple start random hill climbing*).

1.5.2 Le recuit simulé

La méthode du recuit simulé (SA : *Simulated annealing*) trouve ses origines dans le formalisme de mécanique statistique (*algorithme Metropolis* [Metropolis et al., 1953]). Elle a été mise au point par trois chercheurs de la société IBM, S. Kirkpatrick, C.D. Gelatt et M.P. Vecchi en

1983 [Kirkpatrick et al., 1983], et indépendamment par Cerny en 1985 [Cerny, 1985]. La métaheuristique du SA est inspirée du processus de recuit physique utilisé en métallurgie, lui-même reposant sur les lois de thermodynamique énoncées par Boltzmann. Le recuit en métallurgie est un processus visant à réorganiser la structure cristallographique des métaux en alternant des cycles de refroidissement lent et de réchauffage (recuit), qui ont pour effet de minimiser l'énergie du matériau. Chaque température est maintenue jusqu'à ce que le matériau atteigne l'*équilibre thermodynamique*. Le but étant d'obtenir une structure « bien ordonnée » du matériau à l'état solide d'énergie minimale, tout en évitant les structures « métastables », caractéristiques des minima locaux de l'énergie.

Cette méthode est transposée en optimisation pour trouver les extrema d'une fonction : la fonction objectif, assimilée à l'énergie d'un matériau, est alors minimisée, moyennant l'introduction d'une *température fictive*, qui est contrôlée par une fonction décroissante qui définit un schéma de refroidissement. L'algorithme commence par générer une solution initiale (soit de façon aléatoire ou par une heuristique). À chaque nouvelle itération, une solution s' est générée de manière aléatoire dans le voisinage $\mathcal{N}(s)$ de la solution courante s. La solution s' est retenue si elle est de performance supérieure ou égale à celle de la solution courante, i.e., $f(s') \leq f(s)$. Dans le cas contraire, s' est acceptée avec une probabilité $\exp\left(-\frac{\Delta f}{T}\right)$. Cette probabilité dépend de deux facteurs : d'une part l'importance de la dégradation ($\Delta f = f(s') - f(s)$) – les dégradations plus faibles sont plus facilement acceptées ; d'autre part, un paramètre de température T – une température élevée correspond à une probabilité plus grande d'accepter des dégradations. Ainsi, au début de la recherche, la probabilité d'accepter des dégradations est élevée et elle diminue progressivement.

Dans la méthode SA, les mécanismes d'*intensification* et de *diversification* sont contrôlés par la température. La température T ne fait que décroitre pendant le processus de recherche, de sorte que la recherche tend à s'intensifier vers la fin de l'algorithme. L'idée est de diminuer petit à petit la chance d'accepter des solutions qui dégradent la fonction objectif.

1.5.2.1 Les méthodes d'acceptation avec seuil

Les méthodes d'acceptation avec seuil (TA *Threshold accepting method*) sont des variantes du recuit simulé [Dueck & Scheuer, 1990]. La principale différence entre ces deux méthodes se situe au niveau de la détermination du critère d'acceptation d'une solution dégradant la fonction objectif à chaque étape. Dans le SA, ce critère d'acceptation est défini par Metropolis [Metropolis et al., 1953]. Dans une méthode d'acceptation à seuil, la décision d'accepter une dégradation est prise de manière déterministe, en se basant uniquement sur une fonction auxiliaire $\zeta(s, s')$ et sur un seuil S. Dans le cas le plus simple, la fonction $\zeta(s, s')$ est définie par

Δf. Le paramètre de seuil S est défini de manière à diminuer progressivement à chaque fois qu'un nombre prédéterminé d'itérations est effectué. La fonction $\zeta(s, s')$ et le seuil S peuvent être définis de nombreuses manières, ce qui donne lieu à plusieurs variantes pour les méthodes TA.

1.5.3 La méthode de recherche avec tabous

La méthode de recherche avec tabous, ou simplement recherche tabou (TS :*Tabu Search*) a été formalisée par Fred Glover en 1986 [Glover, 1986]. Elle utilise explicitement l'historique de la recherche, à la fois pour échapper aux minima locaux et pour mettre en œuvre une stratégie d'exploration. Sa principale caractéristique est en effet basée sur l'utilisation de mécanismes inspirés de la mémoire humaine. De ce point de vue, la méthode tabou prend un chemin opposé à celui de SA, totalement dépourvu de mémoire, et donc incapable de tirer les leçons du passé.

La méthode TS utilise une mémoire appelée *liste tabou* (d'où le nom attribué à la méthode par Glover), qui enregistre les dernières solutions rencontrées (ou des caractéristiques de solutions) vers lesquelles il est interdit de se déplacer. Ce procédé simple de mémoire permet de choisir le meilleur voisin non tabou, même si celui-ci dégrade la fonction-objectif f. Cependant, dans certains cas, les interdictions occasionnées par la liste tabou peuvent être jugées trop radicales. En effet, on risque d'éliminer (en les rendant tabous), certains mouvements particulièrement utiles. Pour éviter cela, on incorpore dans l'algorithme un *mécanisme d'aspiration* qui détermine des critères selon lesquels un mouvement, bien que tabou, peut quand même être accepté, s'il permet d'obtenir une meilleure solution que toutes celles déjà parcourues.

La taille de la liste tabou contrôle la mémoire du processus de recherche. Pour favoriser l'*intensification*, il suffit de diminuer la taille de la liste tabou. En revanche, augmenter la taille de la liste tabou, forcera le processus de recherche à explorer des régions plus vastes, favorisant ainsi la *diversification*. La taille de la liste tabou peut être modifiée au cours de la recherche [Battiti & Tecchiolli, 1994]. Une autre amélioration intéressante de la TS est l'utilisation de structure de mémoire à moyen et à long terme afin d'approfondir les notions d'intensification et de diversification.

1.5.4 La méthode GRASP

La procédure de recherche gloutonne aléatoire adaptative (GRASP : *Greedy Randomized Adaptive Search Procedure*), proposée par Feo et Resende dans [Feo & Resende, 1989, 1995], est une métaheuristique à départs multiples, dépourvue de mémoire, fondée sur les algorithmes gloutons randomisés et les techniques de recherche de voisinage. Chaque itération de l'algorithme GRASP se compose de deux étapes dites de construction et de recherche locale. L'étape

de la construction est similaire à l'*heuristique semi-gloutonne* (*semi-greedy heuristic*) proposée indépendamment par Hart et Shogan [Hart & Shogan, 1987]. Elle génère une solution réalisable par l'application d'une procédure d'initialisation gloutonne. Dans la deuxième étape, cette solution est utilisée comme solution initiale de la procédure de recherche locale. Après un nombre donné d'itérations, l'algorithme GRASP se termine et la meilleure solution trouvée est conservée. Une étude bibliographique de la méthode GRASP est fournie dans [Festa & Resende, 2009a,b].

1.5.5 La recherche à voisinage variable

La recherche à voisinage variable (VNS : *Variable neighborhood search*) est une métaheuristique proposée par Hansen et Mladenovic en 1997 [Mladenovic, 1995; Mladenovic & Hansen, 1997]. Elle est basée sur le principe de changement systématique de voisinage durant la recherche. À l'étape d'initialisation, un ensemble de structures de voisinage doit être définie. Ces voisinages peuvent être choisis arbitrairement, mais souvent une séquence $\mathcal{N}_1, \mathcal{N}_2, \ldots, \mathcal{N}_{n_{\max}}$ est rangée par ordre de taille croissante des voisinages qu'elle utilise. La procédure de VNS se compose de trois étapes : *perturbation* (*shaking*), *recherche locale* et *déplacement*. Étant donné une solution initiale s, l'étape de perturbation génère au hasard une solution s' dans le $n^{\text{ième}}$ voisinage de la solution courante s. Une procédure de recherche locale est alors appliquée, avec s' comme solution initiale, afin d'obtenir un optimum local s''. Si s'' est meilleure que s, alors s'' remplace s et on génère une nouvelle solution dans le voisinage $\mathcal{N}_1$. Dans le cas contraire, la solution courante reste s et on change de voisinage en générant une solution s' dans le $(n+1)^{\text{ème}}$ voisinage. Plus généralement, on change de voisinage à chaque fois que l'un d'entre eux n'est pas parvenu, après application de la procédure de recherche locale, à améliorer la solution courante s. Par contre, dès qu'un voisinage permet d'améliorer s, alors on recommence le processus avec le premier voisinage.

Cet algorithme est efficace si les structures de voisinage sont complémentaires en ce sens qu'un minimum local pour un voisinage n'en n'est pas nécessairement un pour un autre.

1.5.6 La recherche locale itérée

La recherche locale itérée (ILS :*Iterated local search*) [Stützle, 1998] est une métaheuristique basée sur une idée simple : au lieu de l'application répétée d'une procédure de recherche locale à partir de solutions générées aléatoirement, ILS génère la solution de départ pour la prochaine itération en appliquant une perturbation sur l'optimum local trouvé à l'itération courante. Ceci est fait dans l'espoir que le mécanisme de perturbation fournit une solution située dans le bassin d'attraction d'un meilleur optimum local. Le mécanisme de perturbation est un élément

clé de ILS : d'une part, une perturbation trop faible peut ne pas être suffisante pour s'échapper du bassin d'attraction de l'optimum local actuel ; d'autre part, une perturbation trop forte correspond à une recherche locale multi-départs à partir de combinaisons initiales générées aléatoirement. Un *critère d'acceptation* définit les conditions que le nouvel optimum local p^* doit satisfaire afin de remplacer l'optimum local courant s^*. Le critère d'acceptation, combiné avec le mécanisme de perturbation, permet de contrôler le compromis entre l'intensification et la diversification. Par exemple, un critère d'acceptation qui favorise l'intensification doit accepter seulement les solutions qui sont des améliorations à la valeur généralement optimale. En revanche, le critère qui favorise la diversification accepte toute les solutions perturbées, quelle que soit leur qualité. De nombreux critères d'acceptation qui concilient les deux objectifs peuvent être appliqués [Talbi, 2009].

1.6 Les métaheuristiques à population de solutions

Contrairement aux algorithmes partant d'une solution singulière, les métaheuristiques à population de solutions améliorent, au fur et à mesure des itérations, une population de solutions. On distingue dans cette catégorie, les algorithmes évolutionnaires, qui sont une famille d'algorithmes issus de la théorie de l'évolution par la sélection naturelle, énoncée par Charles Darwin [Darwin, 1859] et les algorithmes d'intelligence en essaim qui, de la même manière que les algorithmes évolutionnaires, proviennent d'analogies avec des phénomènes biologiques naturels.

1.6.1 Les algorithmes évolutionnaires

Les algorithmes évolutionnistes ou algorithmes évolutionnaires (*EC : Evolutionary Computation*), sont une famille d'algorithmes s'inspirant de la théorie de l'évolution « darwinienne » pour résoudre des problèmes divers. Selon la théorie du naturaliste Charles Darwin, énoncée en 1859 [Darwin, 1859], l'évolution des espèces est la conséquence de la conjonction de deux phénomènes : d'une part la *sélection naturelle* qui favorise les individus les plus adaptés à leur milieu à survivre et à se reproduire, laissant une descendance qui transmettra leurs gènes et d'autre part, la présence de variations non dirigées parmi les traits génétiques des espèces (mutations).

Le terme *Evolutionary Computation* englobe une classe assez large de métaheuristiques telles que les algorithmes génétiques [Holland, 1975], les stratégies d'évolution [Rechenberg, 1973], la programmation évolutive [Fogel et al., 1966], et la programmation génétique [Koza, 1992]. La figure 1.3 décrit le squelette d'un algorithme évolutionnaire type, commun à la plupart des instances classiques d'EAs.

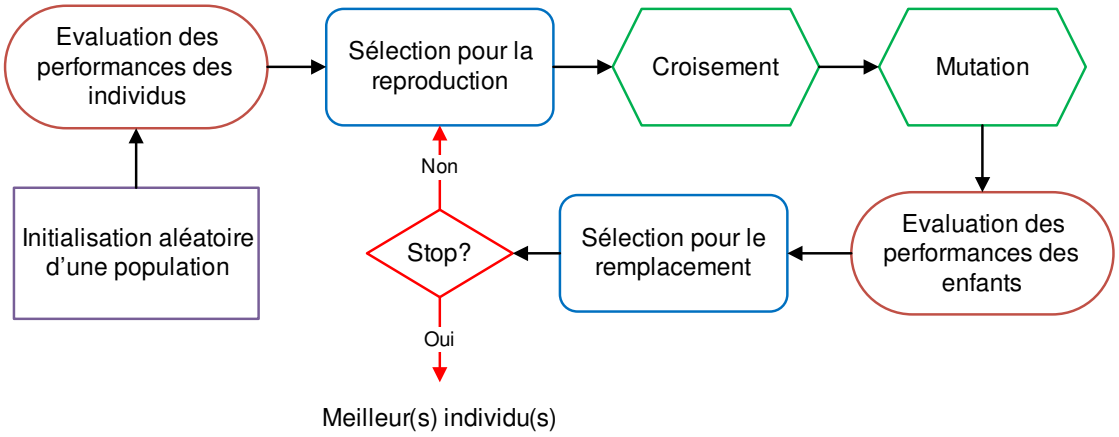


FIGURE 1.3: Principe d'un algorithme évolutionnaire (EA) [Dréo et al., 2003]

Chaque itération de l'algorithme correspond à une *génération*, où une *population* constituée de plusieurs *individus*, représentant des solutions potentielles du problème considéré, est capable de se reproduire. Elle est sujette à des variations génétiques et à la pression de l'environnement qui est simulée à l'aide de la *fonction d'adaptation*, ce qui provoque la *sélection* naturelle (la survie du plus fort). Les opérateurs de variation sont appliqués (avec une probabilité donnée) aux individus *parents* sélectionnés, ce qui génère de nouveaux descendants appelés *enfants* (ou *offsprings*) ; on parlera de *mutation*[4] pour les opérateurs unaires, et de *croisement* pour les opérateurs binaires (ou n-aires). Les individus issus de ces opérations sont alors insérés dans la population. Le processus d'évolution est itéré, de génération en génération, jusqu'à ce qu'une condition d'arrêt soit vérifiée, par exemple, quand un nombre maximum de générations ou un nombre maximum d'évaluations est atteint.

1.6.1.1 Les algorithmes génétiques

Les algorithmes génétiques (GA : *Genetic Algorithms*) sont, sans conteste, la technique la plus populaire et la plus largement utilisée des EAs. Les origines de ces algorithmes remontent au début des années 1970, avec les travaux de John Holland et ses élèves à l'Université du Michigan sur les systèmes adaptatifs [Holland, 1975]. L'ouvrage de référence de David E. Goldberg [Goldberg, 1989] a fortement participé à leur essor. Ces algorithmes se détachent en grande partie par la représentation des données du génotype, initialement sous forme d'un vecteur binaire et plus généralement sous forme d'une chaîne de caractères.

Chaque étape de GA est associée à un opérateur décrivant la façon de manipuler les indi-

4. L'idée directrice de la mutation est de permettre de visiter tout l'espace. Les quelques résultats théoriques de convergence des EAs ont d'ailleurs tous comme condition l'*ergodicité* de la mutation, c'est-à-dire le fait que tout point de l'espace de recherche peut être atteint en un nombre fini de mutations.

vidus :

- *Sélection* : Pour déterminer quels individus sont plus enclins à se reproduire, une sélection est opérée. Il existe plusieurs techniques de sélection, les principales utilisées sont la sélection par tirage à la roulette (*roulette-wheel selection*), la sélection par tournoi (*tournament selection*), la sélection par rang (*ranking selection*), etc [Goldberg & Deb, 1991; Blickle & Thiele, 1995].
- *Croisement* : L'opérateur de croisement combine les caractéristiques d'un ensemble d'individus parents (généralement deux) préalablement sélectionnés, et génère de nouveaux individus enfants. Là encore, il existe de nombreux opérateurs de croisement, par exemple le croisement en un point, le croisement en *n-points* ($n \geq 2$) et le croisement *uniforme*[5] (voir figure 1.4).
- *Mutation* : Les descendants sont mutés, c'est-à-dire que l'on modifie aléatoirement une partie de leur génotype, selon l'opérateur de mutation.
- *Remplacement* : Le remplacement (ou sélection des survivants), comme son nom l'indique, remplace certains des parents par certains des descendants. Le plus simple est de prendre les meilleurs individus de la population, en fonction de leurs performances respectives, afin de former une nouvelle population (typiquement de la même taille qu'au début de l'itération).

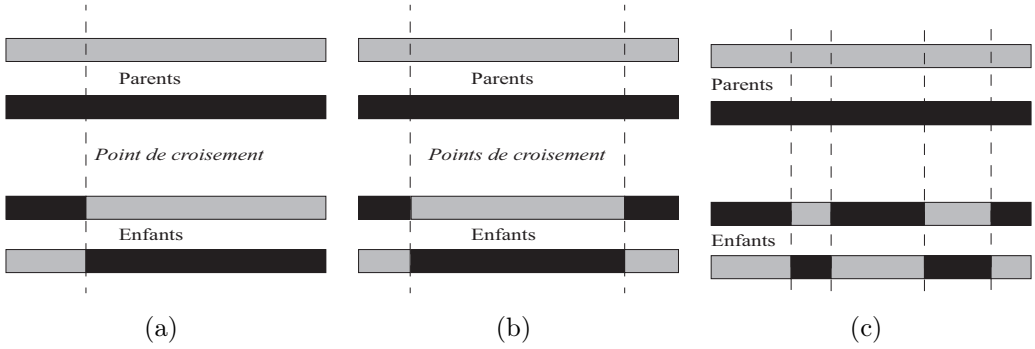


FIGURE 1.4: Exemples de croisement : (a) croisement simple en un point, (b) croisement en deux points, (c) croisement uniforme

1.6.1.2 La Stratégie d'Évolution

La Stratégie d'Évolution (ES : *Evolution Strategy*) a été introduite dans les années 1960 par Rechenberg [Rechenberg, 1965, 1973] et développée par Schwefel. L'algorithme le plus simple, appelé *two membered ES* ou (1 + 1)-ES, manipule un seul individu. A chaque génération,

5. Le croisement uniforme peut être vu comme un croisement multi-points dont le nombre de coupures est déterminé aléatoirement au cours de l'opération.

l'algorithme génère par mutation un individu enfant à partir de l'individu parent et sélectionne l'un ou l'autre pour le conserver dans la population. Pour introduire la notion de population, qui n'a pas vraiment été utilisée que dans la version de base de ES, Rechenberg propose le *multimembered ES* (ou $(\mu + 1)$-ES), où $\mu > 1$ parents peuvent participer à la génération d'un individu enfant. Avec l'introduction de plus d'un parent, la recombinaison a été également introduite dans ces algorithmes. Schwefel [Schwefel, 1981] introduit deux autres versions du *multimembered ES*, selon la procédure de sélection utilisée (i.e. $(\mu + \lambda)$-ES et (μ, λ)-ES). La création d'une nouvelle population consiste à générer λ individus à partir de μ parents et à ne conserver que les μ meilleurs, soit parmi les descendants uniquement (alors il est nécessaire que λ soit supérieur à μ, c'est le schéma (μ, λ)-ES), soit parmi les descendants ainsi que leurs parents (schéma $(\lambda + \mu)$-ES). Deux autres versions de ES connues sont $(\mu/\rho + \lambda)$-ES et $(\mu/\rho, \lambda)$-ES. Le paramètre supplémentaire ρ est le nombre de parents impliqués dans la procréation d'un individu enfant.

Un exemple particulièrement connu est la méthode *Covariance Matrix Adaptation Evolution Strategy* (CMA-ES) dont la caractéristique principale est l'auto-adaptation de la matrice de covariance de la distribution normale utilisée pour la mutation [Hansen et al., 1995].

1.6.1.3 La programmation évolutionnaire

La programmation évolutionnaire (EP :*Evolutionary Programming*) a d'abord été présenté en début des années 1960 par L. J. Fogel comme une approche évolutionnaire de l'intelligence artificielle, pour des problèmes d'apprentissage à partir d'automates à états finis [Fogel et al., 1966]. Plus tard, dans les années 1990, elle a été reprise par D.B. Fogel pour résoudre des tâches plus générales [Fogel, 1991, 1995]. La particularité de ces algorithmes est qu'ils n'utilisent que des opérateurs de mutation et de remplacement, donc pas d'opérateur de croisement.

La méthode EP reste peu utilisée, comparée aux autres algorithmes de la même famille, car elle ressemble à un ES, quoique développé complètement indépendamment, avec, toutefois, l'utilisation fréquente d'un remplacement plus stochastique que déterministe dans lequel, les plus mauvais ont tout de même une (petite) chance de survie [Bäck et al., 1993].

1.6.1.4 La programmation génétique

La programmation génétique (GP : *Genetic Programming*) date du début des années 1990, et elle est née avec les travaux de Koza [Koza, 1992]. Il s'agit d'une méthode automatisée pour la création de programmes informatiques à partir d'un énoncé de haut niveau du problème « *what needs to be done* ». La particularité de la GP repose sur le fait qu'elle travaille sur une représentation des programmes par arbres et utilise des langages de programmation, dont la

syntaxe est simple et telle que des manipulations syntaxiques également simples produisent encore des programmes licites.

La population initiale est construite par tirage aléatoire d'arbres représentant les programmes. Il est souvent spécifié une profondeur limite ou un nombre maximal de nœuds que les arbres générés aléatoirement ne peuvent dépasser. Les trois opérateurs les plus employés en GP sont : la mutation, le croisement et la reproduction. D'autres opérateurs de transformation peuvent être intégrés, comme la *permutation*, l'*édition*, ou l'*encapsulation* [Koza, 1994].

1.6.2 Autres algorithmes évolutionnaires

D'autres méthodes sont également qualifiées d'algorithmes évolutionnaires, bien qu'elles n'en utilisent pas tous les concepts. C'est le cas par exemple des algorithmes à estimation de distribution, les algorithmes à évolution différentielle, les algorithmes coévolutionnaires et les algorithmes culturels.

1.6.2.1 Les algorithmes à estimation de distribution

Les algorithmes à estimation de distribution (EDA : *Estimation of Distribution Algorithms*), aussi appelés algorithmes génétiques par construction de modèles probabilistes (PMBGA : *Probabilistic Model-Building Genetic Algorithms*), ont été proposés en 1996 par Mühlenbein et Paaß [Mühlenbein & Paaß, 1996]. Ils ont une base commune avec les algorithmes évolutionnaires, mais s'en distinguent, car le cœur de la méthode consiste à estimer les relations entre les différentes variables du problème. Ils utilisent pour cela une estimation de la distribution de probabilité, associée à chaque point de l'échantillon. Ils n'emploient donc pas d'opérateurs de croisement ou de mutation, l'échantillon étant directement construit à partir des paramètres de distribution, estimés à l'itération précédente.

Les EDAs peuvent être divisés en trois classes, selon la complexité des modèles probabilistes utilisés pour évaluer les dépendances entre variables [Larrañaga & Lozano, 2002] : (1) les modèles *sans* dépendance (*Univariate* EDAs) ; (2) les modèles à dépendances *bivariantes* (*Bivariate* EDAs) ; et (3) les modèles à dépendances *multiples* (*Multivariate* EDAs).

1.6.2.2 L'évolution différentielle

L'algorithme à évolution différentielle (DE : *Differential Evolution*) a été proposé par R. Storn et K. Price dans les années 1990 [Storn & Price, 1997] afin de résoudre le problème d'ajustement par polynômes de Tchebychev (*Chebyshev polynomial fitting problem*).

Différentes variantes de DE ont été suggérées par Price et al. [Price et al., 2005] et sont classiquement appelées *DE/x/y/z*, où *DE* désigne l'évolution différentielle, x fait référence au

mode de sélection de l'individu de référence ou l'individu cible (*rand* ou *best*) pour la mutation, y est le nombre de différenciations utilisées pour la perturbation du vecteur cible x et z désigne le schéma de croisement, qui peut être *binomial* ou *exponentiel*. À chaque itération du processus d'évolution, chaque individu est d'abord muté, puis croisé avec son mutant. La phase de *mutation* implique que, pour chaque individu de la population $\vec{X}_i$ (*target individual*), un nouvel individu $\vec{V}_i$ (*mutant individual*) est généré, en ajoutant à ses composantes la différence pondérée d'autres individus pris aléatoirement dans la population. Le vecteur d'essai $\vec{U}_i$ (*trial individual*) est ensuite généré après *croisement* entre l'individu initial $\vec{X}_i$ et son mutant $\vec{V}_i$. La phase de *sélection* intervient juste après, par compétition entre l'individu père $\vec{X}_i$ et son descendant $\vec{U}_i$, le meilleur étant conservé pour la génération suivante. Ce processus est répété pour chaque individu de la population initiale, et mène donc à la création d'une nouvelle population de taille identique.

1.6.2.3 Les algorithmes coévolutionnaires

En biologie, on parle de *coévolution* lorsque les évolutions des espèces en interaction – par exemple, prédateurs et proies, hôtes et parasites ou insectes et les fleurs qu'elles pollinisent – s'influencent réciproquement. La coévolution biologique, rencontrée dans de nombreux processus naturels, a été une source d'inspiration pour les algorithmes coévolutionnaires (CoEA : *CoEvolutionary Algorithms*), où deux ou plusieurs populations d'individus, chacune s'adaptant aux changements de l'autre, sont en interaction constante et co-évoluent simultanément, en contraste avec une seule population dans les EAs traditionnels. D'importantes recherches sur les CoEAs ont commencé au début des années 1990 avec les travaux fondateurs de Hillis [Hillis, 1990] sur les réseaux de tri.

De nombreuses variantes des CoEAs ont été mises en œuvre, et elles se répartissent en deux catégories : *les CoEAs concurrentiels* et *les CoEAs coopératifs*. Dans le cas des approches *concurrentielles*, les différentes populations sont en compétition pour la résolution du problème auquel elles sont confrontées et les individus sont récompensés au détriment de ceux avec lesquels ils interagissent [Stanley & Miikkulainen, 2004]. Dans le cas des approches *coopératives*, les diverses populations isolées co-évoluent de concert pour résoudre le problème. Les individus sont récompensés quand ils fonctionnent bien ensemble et punis quand ils échouent ensemble [Potter & De Jong, 1994].

1.6.2.4 Les algorithmes culturels

Le terme *culture* a été introduit par l'anthropologue Edward Burnett Tylor dans son livre, « *Primitive Culture* » [Tylor, 1924]. Dès le début de son ouvrage, Tylor donna une définition de

la culture qui a été par la suite citée de nombreuses fois : « *cet ensemble complexe qui comprend les connaissances, les croyances, l'art, le droit, la morale, les coutumes, et toutes les autres aptitudes et habitudes qu'acquiert l'homme en tant que membre d'une société* ». Les algorithmes culturels (CA : *Cultural Algorithms*), introduits par Robert G. Reynolds, correspondent à des modélisations inspirées de l'évolution de la culture humaine [Reynolds, 1994]. De ce fait, de même que l'on parle d'évolution biologique comme résultat d'une sélection s'exerçant sur la variabilité génétique, on peut parler d'une *évolution culturelle* résultant d'une sélection s'exerçant sur la variabilité culturelle. De cette idée, Reynolds a développé un modèle dont l'évolution culturelle est considérée comme un processus de transmission d'expérience à deux niveaux : un niveau *micro-évolutionnaire* en termes de transmission de matériel génétique entre individus d'une population et un niveau *macro-évolutionnaire* en termes de connaissances acquises sur la base des expériences individuelles.

1.6.3 L'intelligence en essaim

L'intelligence en essaim (SI : Swarm Intelligence) est née de la modélisation mathématique et informatique des phénomènes biologiques rencontrés en éthologie [Bonabeau et al., 1999]. Elle recouvre un ensemble d'algorithmes, à base de population d'agents simples (entités capables d'exécuter certaines opérations), qui interagissent localement les uns avec les autres et avec leur environnement. Ces entités, dont la capacité individuelle est très limitée, peuvent conjointement effectuer de nombreuses tâches complexes nécessaires à leur survie. Bien qu'il n'y ait pas de structure de contrôle centralisée qui dicte la façon dont les agents individuels devraient se comporter, les interactions locales entre les agents conduisent souvent à l'*émergence* d'un comportement collectif global et *auto-organisée*.

Deux exemples phares d'algorithmes de l'intelligence en essaim sont les algorithmes de colonies de fourmis et les algorithmes d'optimisation par essaim particulaire. D'autres algorithmes d'optimisation qui proviennent d'analogies avec des phénomènes biologiques naturels ont été proposés. Parmi les plus significatifs d'entre eux figurent : l'algorithme *Bacterial foraging optimization* [Passino, 2002] qui s'inspire du comportement de recherche de nourriture et de reproduction des bactéries, l'optimisation par colonie d'abeilles (*Bee Colony Optimization*) [Walker et al., 1993], les systèmes immunitaires artificiels [Farmer et al., 1986] et l'algorithme d'optimisation basée sur la biogéographie insulaire *Biogeography-Based Optimization* [Simon, 2008], dont il est question dans cette thèse.

1.6.3.1 Les colonies de fourmis

L'optimisation par colonie de fourmis (ACO : *Ant Colony Optimization*) a été initialement introduite par Marco Dorigo et ses collègues [Dorigo et al., 1991; Dorigo, 1992; Dorigo et al., 1996]. Elle puise son inspiration dans le comportement des fourmis réelles à la recherche de nourriture [Deneubourg et al., 1990]. En effet, celles-ci parviennent à trouver le chemin le plus court entre leur nid et une source de nourriture, sans pour autant avoir des capacités cognitives individuelles très développées. Les fourmis explorent d'abord les environs de leur nid en effectuant une marche aléatoire. Le long de leur chemin entre la source de nourriture et le nid, elles déposent sur le sol une substance chimique volatile appelée *phéromone* afin de marquer certains chemins favorables qui devraient guider leurs congénères à la source de nourriture [Dorigo & Blum, 2005]. Après un certain temps, le chemin le plus court entre le nid et la source de nourriture présente une plus forte concentration de phéromone et, par conséquent, attire plus de fourmis. Les colonies de fourmis artificielles exploitent cette caractéristique pour construire des solutions à un problème d'optimisation. Elles construisent des solutions en parcourant un *graphe de construction* complet $G_C(V,E)$ dont les nœuds V sont des composants de solutions [6], et les arêtes E sont des connexions entre les composants. L'algorithme général de ACO se décompose, pour chaque itération, en trois étapes :

1. *ConstructAntSolutions*. Dans cette étape, un ensemble de m fourmis artificielles construisent itérativement des solutions à partir de l'ensemble des composants de solution possibles $\mathcal{C} = \left\{c_i^j\right\}$ $(i = 1,\ldots,n\,;\; j = 1,\ldots,|D_i|)$. La construction d'une solution commence par une solution partielle vide $s_p = \varnothing$. Puis, à chaque étape de la construction, la solution partielle courante s_p est étendue en y ajoutant un composant de solution c_i^j parmi l'ensemble des voisins réalisables $\mathcal{N}(s_p) \subseteq \mathcal{C}$. Le choix d'un composant dans $\mathcal{N}(s_p)$ se fait de manière probabiliste à chaque étape de la construction. Chaque composant $c_i^j \in \mathcal{N}(s_p)$ a une probabilité $p\left(c_i^j|s_p\right)$ d'être choisie. Cette décision est généralement influencée par la quantité de phéromone τ_{ij} associée à chacun des éléments de $\mathcal{N}(s_p)$, et par l'information heuristique sur le problème. La règle la plus largement utilisée pour ce choix stochastique est celle initialement proposée pour *Ant System* (AS) [Dorigo et al., 1996] et donnée par l'équation (1.6.1).

$$p\left(c_i^j|s_p\right) = \frac{\tau_{ij}^{\alpha}.\left[\eta\left(c_i^j\right)\right]^{\beta}}{\sum_{c_i^l \in \mathcal{N}(s_p)} \tau_{il}^{\alpha}.\left[\eta\left(c_i^l\right)\right]^{\beta}}, \forall c_i^j \in \mathcal{N}(s_p) \tag{1.6.1}$$

L'information heuristique, désignée par la fonction $\eta(.)$, attribue à chaque composant de

6. La définition d'un composant de solution dépend du problème étudié. Dans le problème du voyageur de commerce (TSP : *Traveling Salesman Problem*) [Lawler et al., 1985], un composant de solution est une ville qui est ajoutée à un tour.

solution faisable $c_i^j \in \mathcal{N}(s_p)$ une valeur heuristique. Cette valeur est utilisée par les fourmis pour prendre des décisions probabilistes sur la manière de se déplacer dans le graphe de construction. Les paramètres α et β déterminent l'influence des valeurs de phéromone (resp. des valeurs heuristiques) sur les décisions de la fourmi [Dorigo & Thomas, 2010]. Ils ont pour rôle de contrebalancer l'algorithme dans une phase d'intensification ou de diversification.

2. *DaemonActions*. Les actions *Daemon* font référence aux actions spécifiques au problème ou à des actions centralisées qui ne peuvent pas être effectuées séparément par chaque fourmi. Le plus souvent, ces actions consistent en une recherche locale parmi les solutions construites, et où seules les solutions localement optimisées sont utilisées dans la mise à jour des traces de phéromones.

3. *UpdatePheromones*. Le processus de mise à jour des traces de phéromones s'effectue en deux étapes : une réduction de toutes les valeurs de phéromones par un procédé appelé *évaporation* et l'augmentation des valeurs de phéromones, associées à un ensemble de bonnes solutions choisies S_{upd}, par un procédé appelé *dépôt* de phéromone. La mise à jour des traces de phéromones diffère selon l'algorithme ACO utilisé. Elle est généralement implémentée comme suit [Dorigo & Thomas, 2010] :

$$\tau_{ij} = (1-\rho)\tau_{ij} + \sum_{s \in S_{upd} | c_i^j \in s} g(s) \tag{1.6.2}$$

où $0 < \rho \leq 1$ est le taux d'évaporation des phéromones et $g : \mathcal{S} \longmapsto \mathbb{R}^+$ est une fonction telle que : $f(s) < f(\acute{s}) \Rightarrow g(s) \geq g(\acute{s})$. La fonction g est communément appelé fonction qualité (*quality function*).

1.6.3.2 L'optimisation par Essaim particulaire

L'optimisation par essaim particulaire (PSO : *Particle Swarm Optimization*) a été inventée par Russel Eberhart et James Kennedy en 1995 [Kennedy & Eberhart, 1995]. Elle s'inspire des déplacements collectifs observés chez certains animaux sociaux tels que les poissons et les oiseaux migrateurs qui ont tendance à imiter les comportements réussis qu'ils observent dans leur entourage, tout en y apportant leurs variations personnelles. Elle trouve ses origines dans les travaux de C. Reynolds [Reynolds, 1987] et de Heppner et Grenander [Heppner & Grenander, 1990] qui ont créé des modèles mathématiques permettant de simuler des vols groupés d'oiseaux et de bancs de poissons.

L'essaim particulaire correspond à une population d'agents appelés « particules ». Chaque particule, modélisée comme une solution potentielle au problème d'optimisation, parcourt l'es-

pace de recherche, en quête de l'optimum global. Le déplacement d'une particule est influencé par trois composantes (cf. figure 1.5) : (1) une *composante physique* : la particule tend à suivre sa direction courante de déplacement ; (2) une *composante cognitive* : la particule tend à se diriger vers le meilleur site par lequel elle est déjà passée ; (3) 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.

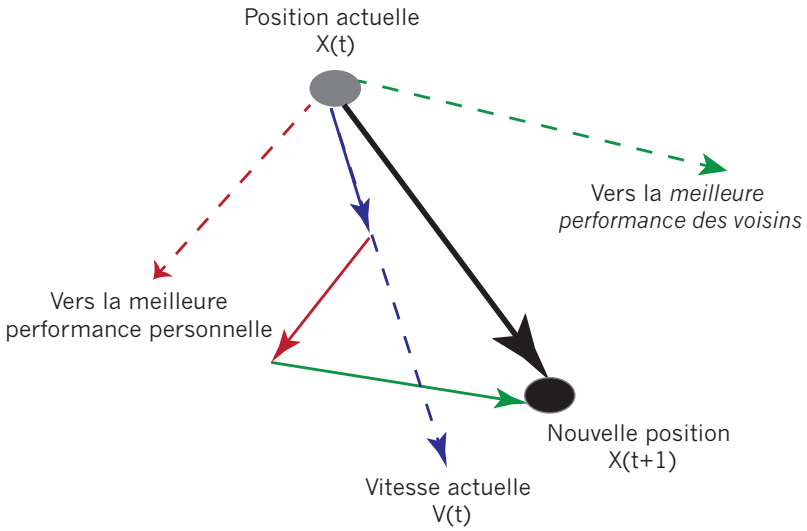


FIGURE 1.5: PSO - Déplacement d'une particule

Le voisinage peut être défini spatialement en prenant par exemple la distance euclidienne entre les positions de deux particules ou sociométriquement, en prenant la position de l'individu dans l'essaim [Kennedy & Mendes, 2002]. Chaque particule i de l'essaim est caractérisée à la fois par sa position $\vec{X}_i$ et par un vecteur de changement de position (appelé *vélocité* ou vitesse) $\vec{V}_i$. Chaque particule dispose d'une mémoire lui permettant de se souvenir de sa meilleure solution, découverte par le passé, que l'on note $\vec{P}_i$ (*personal best*) ainsi que de la meilleure position connue de son voisinage, notée $\vec{P}_g$ (*global best*). À chaque itération, chaque particule se déplace dans l'espace de recherche en suivant un vecteur, calculé comme somme pondérée des vecteurs représentant sa vitesse courante ($\vec{V}_i$), ainsi que sa $\vec{P}_i$ et sa $\vec{P}_g$ (cf. équation (1.6.3) et figure (1.5)). Sa nouvelle vitesse $\vec{V}_i(t+1)$ est déterminée de la façon suivante :

$$\vec{V}_i(t+1) = \omega\vec{V}_i(t) + C_1\varphi_1\left(\vec{P}_i(t) - \vec{X}_i(t)\right) + C_2\varphi_2\left(\vec{P}_g(t) - \vec{X}_i(t)\right) \tag{1.6.3}$$

où $i = 1, 2, \ldots, N$, et N est le nombre de particules (taille de l'essaim) ; le cœfficient d'inertie ω [Shi & Eberhart, 1998] permet de contrôler l'influence de la vitesse obtenue au pas précédent. Un grand facteur d'inertie provoque une grande amplitude de mouvement alors qu'un petit facteur d'inertie concentre la recherche sur un petit espace ; φ_1 et φ_2 sont deux nombres aléatoires tirés uniformément dans [0,1], C_1 et C_2 sont deux constantes qui représentent une

accélération positive. Elles correspondent à la composante *cognitive* (resp. *sociale*) du déplacement. La position de chaque particule est également mise à jour à chaque itération comme suit :

$$\vec{X}_i(t+1) = \vec{X}_i(t) + \vec{V}_i(t+1) \tag{1.6.4}$$

Afin d'éviter que les particules ne se déplacent trop rapidement dans l'espace de recherche, passant éventuellement à côté de l'optimum, il peut être nécessaire de fixer une vitesse maximale (notée $V_{\max}$), de telle sorte que chaque composante de $\vec{V_i}$ soit maintenue dans l'intervalle $[-V_{\max}, +V_{\max}]$ [Eberhart et al., 1996]. Le choix de $V_{\max}$ reste une opération délicate, car ce paramètre a un impact sur la balance exploration/exploitation. L'utilisation d'un cœfficient de constriction χ permet de s'affranchir de la définition de la vitesse maximale V_{max} [Clerc & Kennedy, 2002].

1.6.3.3 Systèmes immunitaires artificiels

Les travaux sur les systèmes immunitaires artificiels (AIS : *Artificial Immune Systems*) ont commencé dans le milieu des années 1980 avec l'article de Farmer, Packard et Perelson [Farmer et al., 1986]. Ils sont inspirés du fonctionnement du système immunitaire humain, dont la fonction la plus importante est de protéger le corps contre les pathogènes (micro-organismes nuisibles tels que les bactéries et les virus). Les principes puisés dans les systèmes immunitaires, comprenant la *reconnaissance de motifs*, l'*hypermutation*, la *sélection clonale*, la *théorie du danger*, la *théorie des réseaux immunitaires*, et bien d'autres, ont inspiré beaucoup de chercheurs dans la conception d'outils d'ingénierie pour résoudre des tâches complexes. La famille des AIS peut être divisée en quatre types d'algorithmes : (1) les algorithmes de sélection négatifs [Forrest et al., 1994], (2) les réseaux immunitaires artificiels [Jerne, 1973], (3) les algorithmes de sélection clonale [de Castro & Von Zuben, 2002] ; (4) la théorie du danger [Aikelin & Cayzer, 2002] et les algorithmes de cellules dendritiques [Greensmith et al., 2005] . Une discussion détaillée de ces algorithmes peut être trouvée dans [Timmis et al., 2008] et [Dasgupta et al., 2011].

1.6.3.4 L'algorithme à base de biogéographie

L'algorithme à base de biogéographie (BBO : *Biogeography-Based Optimization*), développé par Dan Simon en 2008 [Simon, 2008], trouve ses origines dans la *théorie de l'équilibre dynamique* (appelée aussi théorie de la *biogéographie insulaire*), énoncée par MacArthur et Wilson [MacArthur & Wilson, 1967].

La théorie de la biogéographie consiste en l'étude de la répartition spatiale des espèces vivantes (végétales et animales) et des causes de cette répartition. Elle traite de la façon dont

la richesse en espèces (nombre d'espèces) est maintenue dans un système d'île [7] qui sont sujettes à l'immigration et sur lesquelles des espèces s'éteignent [Ricklefs & Miller, 2005]. Elle stipule que les milieux insulaires sont à l'origine vides d'espèces et que celles-ci y arrivent peu à peu en provenance de régions vastes (désignées sous le terme de « continents », bien qu'il ne s'agisse pas forcément de continents à proprement parler) ou d'îles voisines. Certaines espèces sont d'ailleurs mieux outillées que d'autres pour conquérir de nouveaux territoires, elles ont donc des capacités de colonisation des milieux insulaires plus grandes que d'autres. Les interactions compétitives sur l'île tendent par contre à accélérer les extinctions. Le croisement de ces deux processus dynamique permet d'expliquer la richesse actuelle du peuplement. À l'équilibre, il y a un remplacement constant des espèces.

L'algorithme BBO manipule une population d'individus appelés *îles* (ou *habitats*). Chaque île représente une solution possible au problème à résoudre. La « *fitness* » de chaque île est déterminée par son HSI (*Habitat Suitability Index*), une mesure de la qualité d'une solution candidate, et chaque île est représentée par des SIVs (*Suitability Index Variables*). Une bonne solution au problème d'optimisation est une île avec un grand nombre d'espèces, ce qui correspond à une île avec un faible *HSI*.

Selon la théorie de MacArthur et Wilson, le nombre d'espèces présentes sur une île dépend essentiellement d'un équilibre entre le taux d'immigration de nouvelles espèces et le taux d'émigration [8] des espèces déjà établies sur l'île. Dans BBO, chaque habitat a son propre taux d'immigration (λ) – arrivées venant de l'extérieur – et son taux d'émigration (μ) – départs vers l'extérieur. Ces paramètres sont influencés par le nombre d'espèces (S) présentes sur l'île.

Le taux d'immigration (λ) décroît avec l'augmentation du nombre d'espèces (S) déjà présentes sur l'île. Plus le nombre d'espèces déjà installées sur l'île augmente, de moins en moins d'immigrants appartenant à une nouvelle espèce rejoignent l'île. Mais, au fur et à mesure que le nombre d'espèces déjà présentes sur l'île diminue, plus le taux d'immigration augmente. Le taux d'immigration maximale (I) est atteint lorsque l'île est vide. Une fois que toutes les espèces sont présentes sur l'île, alors $S = S_{\max}$ (capacité maximale de l'île) et le taux d'immigration tombe à zéro, ne favorisant plus l'installation de nouveaux arrivants (plus l'île est peuplée, moins les espèces étrangères ont de chances de s'y implanter). Le taux d'immigration, quand il y a S espèces sur l'île, est donné par :

$$\lambda_S = I\left(1 - \frac{S}{S_{max}}\right) \tag{1.6.5}$$

7. Ce que cette approche désigne sous le vocable d'« île » n'est pas nécessairement une île au sens propre du terme. Les lacs peuvent être assimilés à des milieux insulaires, de même que des fragments d'habitat isolés. La théorie de la biogéographie insulaire a d'ailleurs été étendue aux péninsules, aux baies et à d'autres régions qui ne sont isolées que partiellement.

8. Pour l'algorithme BBO, on utilise le terme émigration à la place du terme extinction.

Le taux d'émigration (μ) augmente avec le nombre d'espèces (S) présentes sur l'île. Le taux d'émigration maximum (E) se produit lorsque toutes les espèces sont présentes sur l'île ($S = S_{\max}$), et devient nul si les espèces présentes sur l'île s'éteignent (ou quittent l'île). Le taux d'émigration quand il y a S espèces sur l'île est donné par :

$$\mu_S = E\left(\frac{S}{S_{max}}\right) \tag{1.6.6}$$

La figure 1.6 représente graphiquement le modèle d'équilibre du nombre d'espèces sur les îles. Le nombre d'espèces déjà établies sur une île a un effet négatif sur l'immigration (compétiteurs, prédateurs et parasites déjà présents, moins d'espèces qui restent à immigrer), et un effet positif sur l'émigration (moins de ressources par espèce, forte compétition interspécifique). Le taux d'immigration chute rapidement au début lorsque les meilleurs colonisateurs s'établissent sur l'île. Le taux d'émigration s'accroît plus rapidement avec un nombre élevé d'espèces déjà présentes sur l'île. Le nombre d'espèces à l'équilibre sur l'île (S^*) est déterminé par l'intersection des courbes d'émigration (E) et d'immigration (I). Le modèle de la figure 1.6 représente l'évolution du taux d'immigration (resp. d'émigration) par une fonction linéaire décroissante (resp. croissante) du nombre d'espèces présentes sur l'île. Il existe toutefois différents modèles mathématiques de la biogéographie qui comprennent des variables plus complexes [MacArthur & Wilson, 1967]. Il y a, en effet, d'autres facteurs importants qui influencent les taux de migration entre les habitats, tels que la distance entre les habitats, la taille de l'habitat, variations climatiques (pluviométrie, température), la diversité végétale et animale, en plus de l'activité humaine. Ces facteurs rendent les courbes d'immigration et d'émigration plus complexes, contrairement à celles décrites dans le document original sur BBO [Simon, 2008]. Pour examiner l'influence de différents modèles de migration sur les performances de BBO, Haiping Ma [Ma, 2010] a étudié le comportement de six modèles de migration. Les résultats expérimentaux montrent clairement que les modèles de migration les plus proches de la nature (c'est-à-dire, non-linéaires) sont nettement mieux que les modèles linéaires.

Considérons à présent la probabilité P_S que l'île abrite exactement S espèces. Le nombre des espèces change pendant l'intervalle de temps $[t, t + \Delta t[$ selon l'équation suivante :

$$P_S(t + \Delta t) = P_S(t)(1 - \lambda_S \Delta t - \mu_S \Delta t) + P_{S-1}\lambda_{S-1}\Delta t + P_{S+1}\mu_{S+1}\Delta t \tag{1.6.7}$$

L'équation (1.6.7) stipule que le nombre des espèces sur l'île dépend du nombre total des espèces déjà établies sur l'île, de la fréquence à laquelle les nouvelles espèces arrivent et de la fréquence à laquelle les anciennes disparaissent. Nous supposons ici que Δt est assez petit pour que la probabilité que deux changements ou plus se produisent pendant un tel intervalle est nulle. Afin de disposer de (S) espèces à l'instant $(t + \Delta t)$, l'une des conditions suivantes doit être remplie :

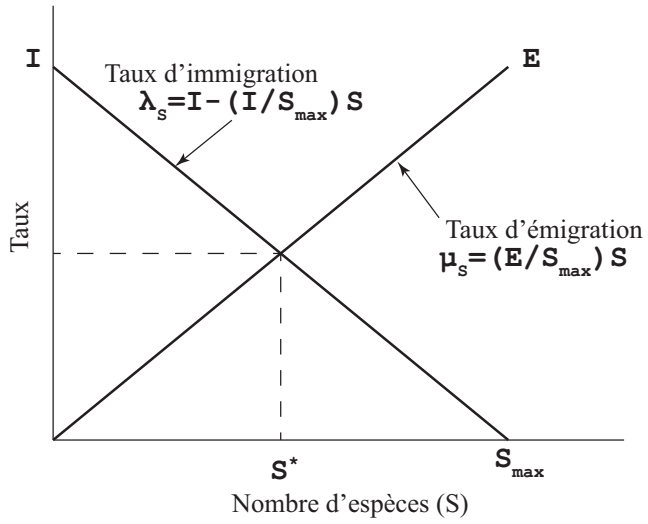


FIGURE 1.6: Modèle de migration linéaire - relations entre le nombre d'espèces (S), le taux d'émigration (μ) et le taux d'immigration (λ)

- Il y a S espèces à l'instant t, et aucune immigration ni émigration n'a eu lieu entre l'instant t et l'instant $t + \Delta t$;
- Il y a $S - 1$ espèces sur l'île à l'instant t, et une nouvelle espèce s'y installe.
- Il y a $S + 1$ espèces sur l'île à l'instant t, et une espèce quitte l'île.

La limite de (1.6.7) quand $\Delta t \to 0$ est donnée par l'équation (1.6.8).

$$\dot{P}_S = \begin{cases} -(\lambda_S + \mu_S)P_S + \mu_{S+1}P_{S+1} & \text{si } S = 0 \\ -(\lambda_S + \mu_S)P_S + \lambda_{S-1}P_{S-1} + \mu_{S+1}P_{S+1} & \text{si } 1 \leq S \leq S_{max} - 1 \\ -(\lambda_S + \mu_S)P_S + \lambda_{S-1}P_{S-1} & \text{si } S = S_{max} \end{cases} \tag{1.6.8}$$

On peut écrire l'équation (1.6.8) sous forme matricielle :

$$\begin{bmatrix} \dot{P}_0 \\ \dot{P}_1 \\ \vdots \\ \vdots \\ \dot{P}_n \end{bmatrix} = \begin{bmatrix} -(\lambda_0 + \mu_0) & \mu_1 & 0 & \dots & 0 \\ \lambda_0 & -(\lambda_1 + \mu_1) & \mu_2 & \dots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \lambda_{n-2} & -(\lambda_{n-1} + \mu_{n-1}) & \mu_n \\ 0 & \dots & 0 & \lambda_{n-1} & -(\lambda_n + \mu_n) \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ \vdots \\ \vdots \\ P_n \end{bmatrix} \tag{1.6.9}$$

Par souci de concision dans les notations, nous écrivons simplement $n = S_{max}$.

L'algorithme BBO peut être décrit globalement par l'algorithme 1.1. Les deux opérateurs de base qui régissent le fonctionnement de BBO sont *la migration* et la *mutation*. En plus, une stratégie d'élitisme est adoptée dans l'algorithme BBO, afin de garder dans la nouvelle population la meilleure solution.

Algorithme 1.1: BBO

Générer aléatoirement un ensemble de solutions initiales (îles)
tant que le critère d'arrêt n'est pas atteint **faire**
    Évaluer la *fitness* (HSI) de chaque solution
    Calculer le nombre d'espèce S, le taux d'immigration λ et d'émigration μ pour chaque solution
    Migration :
    pour $i = 1$ à N **faire**
        Utiliser λ_i pour décider, de manière probabiliste, d'immigrer à $\vec{X}_i$
        si $rand(0,1) < \lambda_i$ **alors**
            pour $j = 1$ à N **faire**
                Sélectionner l'île d'émigration $\vec{X}_j$ avec une probabilité $\propto \mu_j$
                si $rand(0,1) < \mu_j$ **alors**
                    Remplacer une variable de décision (SIV) choisie aléatoirement dans $\vec{X}_i$ par la variable correspondante dans $\vec{X}_j$
                fin
            fin
        fin
    fin
    Mutation : muter les individus au taux de mutation donné par l'équation (1.6.10)
    Remplacement de la population par les descendants
    Implémenter l'élitisme
fin
retourner la meilleure solution trouvée

L'idée générale de la migration est l'échange de caractéristiques entre les îles. Les taux d'immigration (λ) et d'émigration (μ) de chaque île sont utilisés pour transmettre, de manière probabiliste, les caractéristiques entre les îles, $rand(0,1)$ est un nombre aléatoire uniformément distribué dans l'intervalle $[0,1]$ et $X_{i,j}$ est le $j^{\text{ème}}$ SIV de la solution $\vec{X}_i$. La stratégie de migration de BBO est similaire à la recombinaison des stratégies d'évolution (ES) [Bäck, 1996], dans laquelle plusieurs parents sont recombinés entre eux pour former un unique enfant. La principale différence réside dans le fait que la recombinaison est utilisée pour créer de nouvelles solutions, tandis que la migration est utilisée pour modifier des solutions existantes.

Le *HSI* d'une île peut changer brusquement, en raison d'événements aléatoires : des catastrophes naturelles (tempêtes, ouragans, incendies, ...) ou des épidémies, etc. BBO modélise ce phénomène comme une *mutation des SIVs*, et utilise les probabilités de nombre d'espèces (*species count probabilities* P_S) pour déterminer les taux de mutation. La mutation est utilisée pour améliorer la diversité de la population, empêchant ainsi la recherche de stagner. La proba-

bilité, qu'une solution donnée S, existe *a priori* comme une solution pour le problème considéré, est spécifiée par la probabilité du nombre d'espèces (P_S). Si une île S est sélectionnée pour la mutation, alors une variable *SIV* est modifiée de façon aléatoire en fonction de sa probabilité P_S. Dans ce contexte, il convient de remarquer que des solutions avec des valeurs de *HSI* très élevées ou très faibles ont une faible probabilité d'exister. Tandis que les solutions avec un *HSI* moyen sont relativement probables. Si une solution donnée a une probabilité faible, elle est susceptible d'être mutée à une autre solution. A l'inverse, une solution avec une forte probabilité est en revanche moins susceptible d'être mutée. Le taux de mutation $m(S)$ est inversement proportionnel à P_S :

$$m(S) = m_{max}\left(1 - \frac{P_S}{P_{max}}\right) \tag{1.6.10}$$

où m_{max} est un paramètre défini par l'utilisateur, et $P_{max} = \max_S P_S$, $S = 1, \ldots, S_{max}$. Si une île est sélectionnée pour la mutation, alors un SIV choisi au hasard dans l'île est simplement remplacé par une variable aléatoire générée dans son intervalle de définition.

1.7 Optimisation multiobjectif

Toutes les métaheuristiques présentées ci-dessus concernent le cas de problèmes monoobjectif. Cependant, la plupart des problèmes réels sont en fait des problèmes *multiobjectifs*, c'est-à-dire que l'on cherche à optimiser simultanément plusieurs critères qui sont généralement conflictuels (par exemple, on peut avoir besoin de maximiser la qualité d'un produit tout en minimisant son prix de revient). L'optimisation multiobjectif (dites aussi multicritère) consiste donc à optimiser simultanément plusieurs fonctions. Elle trouve ses racines au cours du $XIX^{\text{ème}}$ siècle dans les travaux en économie d'Edgeworth [Edgeworth, 1885] et de Pareto [Pareto, 1896].

On peut formaliser un problème d'optimisation multiobjectif comme suit :

$$\begin{cases} \min F(x) = (f_1(x), f_2(x), \ldots, f_m(x)) \\ \text{sous la contrainte } x \in C \end{cases} \tag{1.7.1}$$

où $x = (x_1, \ldots, x_n)$ est le vecteur représentant les variables de décision ; C représente l'ensemble des solutions réalisables associé à des contraintes d'égalité, d'inégalité et des bornes explicites (*espace de décision*), et $F(x) = (f_1(x), f_2(x), \ldots, f_m(x))$ est le vecteur de fonctions objectifs à optimiser (ou critères de décision) avec $m \geq 2$ le nombre de fonctions objectifs. L'image de l'espace de décision par la fonction objectif F (ou ensemble des points réalisables $Y = F(C)$) est appelée *espace des objectifs* (ou *espace des critères*). On impose sur cet ensemble une relation d'ordre partiel appelée *relation de dominance*.

Dans la littérature, nous rencontrons deux classifications différentes des méthodes de résolution des problèmes d'optimisation multiobjectif. Le premier classement adopte un point de

vue décideur. Il divise ces méthodes en trois familles suivant la coopération entre la méthode d'optimisation et le décideur :

- Les méthodes *a priori* : Le décideur intervient en amont de la méthode d'optimisation. Il défini ses préférences en pondérant par exemple les objectifs selon leurs importances.
- Les méthodes *progressives* ou *interactives* : Le décideur affine son choix, en écartant ou en favorisant des solutions, au fur et à mesure du déroulement du processus de résolution.
- Les méthodes *a posteriori* : Le décideur intervient en aval de l'optimisation. Il choisit la solution qui lui semble la plus intéressante parmi l'ensemble des solutions fournies par la méthode d'optimisation.

La deuxième classification divise les méthodes suivant leur façon de traiter les fonctions objectifs. On distingue les méthodes agrégées, les méthodes basées sur l'équilibre de Pareto et les méthodes non agrégées et non Pareto, que nous détaillons dans la suite.

1.7.1 Méthodes agrégées

Les méthodes agrégées transforment un problème multiobjectif en un problème monoobjectif, en regroupant les critères à optimiser dans une unique fonction objectif. Les méthodes d'agrégation les plus simples utilisent une fonction de mise à l'échelle de chaque critère, afin de pouvoir les additionner (*modèle additif*) ou bien les multiplier (*modèle multiplicatif*).

La méthode d'agrégation la plus connue et la plus employée est la *moyenne pondérée*. Cette méthode consiste à additionner tous les objectifs en affectant un cœfficient de poids. Ce cœfficient traduit l'importance relative que le décideur attribue à l'objectif :

$$F(x) = \sum_{i=1}^{m} \omega_i f_i(x) \tag{1.7.2}$$

où les poids $\omega_i \geq 0$ sont tels que $\sum_{i=1}^{m} \omega_i = 1$.

Quoi que largement utilisée, cette méthode présente toutefois quelques problèmes. D'un côté, le décideur doit déterminer *a priori* les poids de chaque critère. Ceci repose sur la connaissance du problème par le décideur. D'un autre côté, il doit pouvoir exprimer l'interaction entre les différents critères. Ce problème est d'autant plus délicat qu'il existe plusieurs types d'interaction [Marichal, 2002]. De plus, dans le cas où le front de Pareto (cf. section 1.7.2) est non convexe, certaines solutions peuvent ne pas être accessibles en utilisant cette méthode.

Il existe également d'autres méthodes agrégées. Nous pouvons citer à titre d'exemple *Goal programming* [Charnes & Cooper, 1967], le *min-max* [Coello et al., 1995] et la méthode ϵ-contrainte [Ritzel et al., 1994],

1.7.2 Méthodes basées sur l'équilibre de Pareto

Ces méthodes reposent sur le postulat de V. Pareto [Pareto, 1896] : « *Il existe un équilibre tel que l'on ne peut pas améliorer un critère sans détériorer au moins un des autres critères* ». La notion de solution optimale unique dans l'optimisation monoobjectif disparait pour les problèmes d'optimisation multiobjectif au profit de la notion d'ensemble de solutions *Pareto optimales*, selon le critère de *dominance* au sens de Pareto. On appellera *front de Pareto* (ou *surface de compromis*) du problème, l'ensemble des points de l'espace de recherche tel qu'il n'existe aucun point qui est strictement meilleur que les autres sur tous les critères simultanément. Il s'agit de l'ensemble des meilleurs compromis réalisables entre les objectifs contradictoires. L'objectif de l'optimisation va être d'identifier cet ensemble de compromis optimaux entre les critères. La Figure 1.7 donne un exemple de front de Pareto. Notez cependant que l'on n'est pas toujours dans une situation aussi régulière et que le front de Pareto peut être concave, discontinu, etc. Les solutions placées sur le front de Pareto ne peuvent pas être comparées, aucune n'étant systématiquement meilleure que les autres sur tous les objectifs. C'est le décideur qui aura pour rôle de choisir la solution à retenir.

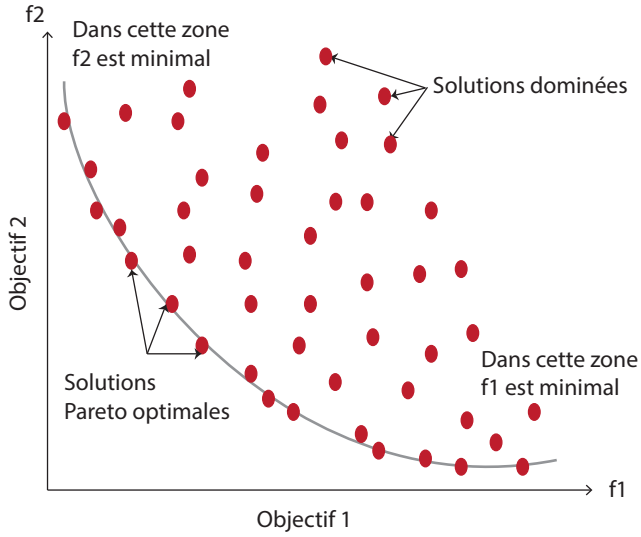


FIGURE 1.7: Front de Pareto et rangs de Pareto pour un problème de minimisation de deux objectifs.

De nombreux algorithmes génétiques ont été développés pour résoudre ce type de problème. Parmi les plus significatifs d'entre eux figurent : l'algorithme Multi Objective Genetic Algorithm (MOGA) [Fonseca & Fleming, 1993], l'algorithme *Niched Pareto Genetic Algorithm* (NPGA) [Horn et al., 1994], utilisant une sélection par tournoi, basée principalement sur la dominance de Pareto, l'algorithme *Non Dominated Sorting Genetic Algorithm* (NSGA) [Srinivas & Deb, 1994] et enfin l'algorithme NSGA-II [Deb et al., 2000] qui est sans doute l'algorithme

le plus populaire.

1.7.3 Méthodes non agrégées et non Pareto

En général, les méthodes non agrégées et non Pareto possèdent un processus de recherche qui traite séparément les objectifs. Elle n'utilisent donc aucun des deux principes énoncés précédemment. L'algorithme génétique à évaluation vectorielle (VEGA : *Vector Evaluated Genetic Algorithm*), présenté par Schaffer en 1985 [Schaffer, 1985], en est un exemple. La seule différence avec un algorithme génétique simple est la manière dont s'effectue la sélection. Si nous avons m objectifs et une population initiale de n individus, alors la population est divisée en m sous-populations, chacune de ces sous-populations contenant les n/m meilleurs individus pour un objectif particulier. La population est ensuite reconstruite et l'on applique les opérateurs génétiques de croisement et de mutation.

1.8 Optimisation sous contraintes

Les problèmes posés dans le monde réel sont souvent soumis à des contraintes. Plusieurs méthodes, classiques et évolutionnaires, ont été développées pour prendre en compte les contraintes. Le problème général d'optimisation sous contraintes peut être formulé comme suit :

$$\begin{cases} \text{Trouver } x \text{ qui optimise} & f(x) \\ \text{sous les contraintes} & g_i(x) \leq 0, \quad i = 1, 2, \ldots, p \\ & h_j(x) = 0, \quad j = 1, 2, \ldots, q \end{cases} \tag{1.8.1}$$

Le vecteur $x = (x_1, \ldots, x_n)$ est le vecteur des n *variables de décision* ($x \in \mathcal{F} \subseteq \mathcal{S} \subseteq \mathbb{R}^n$), avec $\mathcal{F}$ l'espace des solutions réalisables du problème et $\mathcal{S}$ l'espace de recherche. Les contraintes sont des fonctions qui vont restreindre l'espace de recherche. Elles vérifieront si la solution est réalisable, mais elles ne mesureront pas la qualité de cette solution. Il en existe de deux types : contraintes d'inégalité $g_i(x) \leq 0$, $i = 1, 2, \ldots, p$ et contraintes d'égalité $h_j(x) = 0$, $j = 1, 2, \ldots, q$. Si une solution x ne satisfait pas au moins une des contraintes, elle est dite *solution infaisable*, contrairement aux *solutions faisables*, qui vérifient l'ensemble des $p + q$ contraintes. L'ensemble des solutions faisables constitue le *domaine admissible* de l'espace de recherche ou encore l'*espace faisable* $\mathcal{F}$. La recherche de l'optimum faisable est d'autant plus difficile que la taille de l'espace faisable est petite et sa forme est complexe (par exemple des petites régions dispersées). Le rapport entre la taille de l'espace faisable et celle de l'espace de recherche $|\mathcal{F}| \,/\, |\mathcal{S}|$ peut être utilisé comme un indice de difficulté du problème [Michalewicz, 1994].

Plusieurs techniques ont été conçue pour résoudre le problèmes d'optimisation sous contraintes. Nous nous intéressons dans cette section plus particulièrement aux algorithmes évolutionnaires. Chacune des approches développées appartient à l'une des catégories suivantes, pour plus de détail voir [Michalewicz & Schoenauer, 1996; Coello & Becerra, 2002] : (1) Les fonctions de pénalités qui dégradent le résultat de la fonction d'évaluation de l'individu lorsque celui-ci ne respecte pas les contraintes ; (2) les méthodes basées sur la recherche des solutions faisables ; (3) Les méthodes basées sur la préservation de la faisabilité des solutions ; et (4) les méthodes hybrides.

1.8.1 Méthodes basées sur le concept de pénalité

Les méthodes basées sur le principe de pénalisation sont probablement les plus largement utilisées dans la pratique. L'intérêt de ces méthodes est la simplicité de leur mise en œuvre et leur relative efficacité pratique. Elles consistent à transformer le problème (1.8.1) sous contraintes en un problème d'optimisation sans contrainte 1.8.2 en associant à la fonction objectif une pénalité dès qu'une contrainte est violée. La fonction objectif du problème (1.8.1) est alors remplacée par la fonction suivante à optimiser :

$$Fitness(x) = f(x) + F_{penalty}(x) \tag{1.8.2}$$

La fonction à optimiser doit non seulement mesurer la valeur des solutions admissibles, mais également pénaliser les solutions non réalisables. L'ordre de grandeur de la pénalité à accorder à une violation de contrainte n'est pas simple à déterminer. Plusieurs techniques avec des approches différentes ont été proposées dans la littérature. Les plus populaires utilisent la mesure de violation de contraintes pour définir la pénalité. Nous pouvons donc écrire la *fonction de pénalité* sous la forme suivante :

$$F_{penalty}(x) = F\left(\sum_{j=1}^{m} \alpha_j \delta_j^{\beta}(x)\right) \tag{1.8.3}$$

où $\delta_j(x)$ est la mesure du degré de violation de la $j^{\text{ème}}$ contrainte ($j = 1, \ldots, m$ avec $m = p + q$ est le nombre total de contraintes) ; les α_j sont les cœfficients de pénalisation, toujours positifs ; le paramètre β prend généralement les valeurs 1 ou 2. Notons que pour x vérifiant toutes les contraintes (i.e., $x \in \mathcal{F}$), $F_{penalty}(x) = 0$. Le degré de violation de chaque contrainte est calculé comme suit :

$$\delta_j(x) = \begin{cases} \max[0, g_j(x)] & j = 1, \ldots p \\ |h_j(x)| & j = p+1, \ldots, m \end{cases} \tag{1.8.4}$$

Les techniques qui appartiennent à cette catégorie diffèrent dans leur façon de définir la fonction de pénalité. Loin d'être exhaustifs, nous en citons ici quelque exemples.

1.8.1.1 Pénalités statiques

Cette méthode, proposée par *Hamaïfar, Lai* et *Qi* [Homaifar et al., 1994], définit, pour chaque contrainte, une famille d'intervalles de violation (ou taux de violation) déterminant chacun un cœfficient de pénalité approprié. La fonction de pénalité est donnée par :

$$F_{penalty}(x) = \sum_{j=1}^{m} R_{ij}\delta_j^2(x) \tag{1.8.5}$$

où les R_{ij} sont les cœfficients de pénalité associés à la $j^{\text{ème}}$ contrainte. L'indice i représente un intervalle appartenant à une famille d'intervalles de violation, défini *a priori* par l'utilisateur. Le grand inconvénient de cette méthode est le nombre de paramètres à définir.

Une autre approche intéressante basée sur la pénalisation statique a été proposé par *Kuri et al.* [Kuri-Morales & Gutiérrez-García, 2001]. La fonction de pénalité est définie comme suit :

$$F_{penalty}(x) = K\sum_{j=1}^{s}\frac{K}{m} \tag{1.8.6}$$

où s est le nombre de contraintes satisfaites, m est le nombre total de contraintes (d'égalité et d'inégalité), et K est une constante. Nous pouvons noter que les individus qui violent le même nombre de contraintes se voient attribuer le même degré de pénalisation

1.8.1.2 Pénalités dynamiques

Dans les méthodes basées sur la pénalisation dynamique, les cœfficients de pénalité augmentent au fur et à mesure de l'évolution [Joines & Houck, 1994]. La pénalité est calculée comme suit :

$$F_{penalty}(x) = (C \times t)^{\alpha}\sum_{j=1}^{m}\delta_j^{\beta}(x) \tag{1.8.7}$$

où C, α et β son des constantes ($C = 0{,}5$, $\alpha = \beta = 2$ [Joines & Houck, 1994]). Le but est d'augmenter la pression sur l'algorithme pour trouver des solutions admissibles. Cependant cette pression peut parfois être trop forte et gêner le processus d'exploration.

1.8.1.3 Pénalités analytiques

Cette méthode est utilisée dans le système GenocopII (*GEnetic algorithme for Numerical Optimization of COnstrained Problems*). Le système s'inspire de la stratégie de la température

de refroidissement du recuit simulé pour la prise en compte des contraintes non linéaires [Koziel & Michalewicz, 1999]. La fonction de pénalisation est donnée par :

$$F_{penalty}(x,\tau) = \frac{1}{2\tau}\sum_{j=1}^{m}\delta_j^2(x) \tag{1.8.8}$$

où le paramètre τ désigne la température du système, qui diminue à chaque génération selon un schéma de refroidissement choisi par l'utilisateur. L'algorithme s'arrête quand τ atteint la température minimale τ_f choisie préalablement. L'efficacité de la méthode est conditionnée par l'ajustement de la température initiale τ_0, de la température finale τ_f, et du schéma de refroidissement.

1.8.1.4 Pénalités adaptatives

La particularité des méthodes basées sur les pénalités adaptatives est la prise en compte de l'état du processus de recherche, à une génération donnée, dans la définition de la fonction de pénalité. Ainsi, le poids de la pénalité est adapté à chaque itération et il peut être augmenté ou diminué selon la qualité des solutions trouvées. Parmi les méthodes basées sur les pénalités adaptatives, nous pouvons citer la méthode de *Hadj-Alouane* et *Bean* [Hadj-Alouane & Bean, 1992] et la méthode de *Smith* et *Tate* [Smith & Tate, 1993].

1.8.1.5 La pénalité mortelle

La pénalité mortelle (ou *death penalty*) consiste à rejeter de la population les individus non réalisables en attribuant à la fonction objectif une valeur très élevée en cas de minimisation, ou une valeur nulle en cas de maximisation [Bäck et al., 1991]. Elle peut être vue comme une méthode de pénalisation avec une pénalité infinie pour tout x infaisable :

$$F_{penalty}(x) = +\infty \tag{1.8.9}$$

Notons que cette méthode ne peut fonctionner que s'il y a assez d'individus faisable dans la population initiale.

1.8.2 Recherche des solutions faisables

Deux types de méthodes, présentées ci-après, se rattachent à cette catégorie : la réparation des individus infaisables et l'échantillonnage de l'espace faisable. L'objectif de ces deux méthodes est de ramener les individus infaisables dans le domaine réalisable.

1.8.2.1 Réparation des individus infaisables

La méthode de réparation des individus infaisables se base sur l'idée suggérée par *Michalewicz* et *Nazhiyath* en 1995 [Michalewicz & Nazhiyath, 1995] (Genecop III). Elle consiste à faire évoluer deux populations : la première contient les points qui satisfont les contraintes linéaires du problèmes, appelés *points de recherche* ; et la deuxième comporte les individus faisables qui satisfont toutes les contraintes du problème (linéaires et non linéaires), et ils sont appelés *points de référence*. Les points de référence, étant faisables, sont évalués directement en utilisant la fonction objectif. Les points de recherche, quant à eux, sont « réparés » (rendu faisable par une procédure spatiale) avant d'être évalués. L'avantage de cette méthode est que la fonction objectif n'est évaluée que dans l'espace réalisable. Par contre, elle présente des difficultés à créer la population de points de référence si le rapport $|\mathcal{F}| / |\mathcal{S}|$ est très petit.

1.8.2.2 Méthode de la mémoire comportementale

La méthode de la mémoire comportementale (*Behavior memory*) a été proposée par *Schoenauer* et *Xanthakis* en 1993 [Schoenauer & Xanthakis, 1993]. Le principe de cette méthode est d'échantillonner l'espace faisable en traitant les contraintes une par une suivant un ordre particulier. Pour chaque contrainte, l'algorithme fait évoluer la population jusqu'à ce qu'un certain pourcentage de la population devienne faisable pour la contrainte en cours, tout en restant faisable pour les contraintes précédentes. Les individus infaisables sont évalués à la dernière étape en utilisant le principe de la méthode de la peine de mort (*death penalty*) (cf. section 1.8.1.5). La procédure d'échantillonnage nécessite toutefois le choix d'un ordre linéaire pour le traitement des différentes contraintes du problème. Ce choix a une grande influence sur la qualité des résultats.

1.8.3 Préservation de la faisabilité des solutions

Le principe des méthodes appartenant à cette catégorie est de conserver la faisabilité de la population. Elles consistent à générer à partir d'individus faisables, d'autres qui sont aussi faisables en utilisant des opérateurs de reproduction spécifiques.

1.8.3.1 GENOCOP

L'algorithme GENOCOP, développé en 1991 par Michalewicz et Janikow [Michalewicz & Janikow, 1991], traite les problèmes d'optimisation non linéaires à contraintes linéaires. Le principal inconvénient de cette méthode réside dans son incapacité de traiter des contraintes non linéaires, problème qui a été traité dans la deuxième version du système :GenocopII (cf. section 1.8.1.3).

1.8.3.2 La recherche sur la frontière de la région faisable

Michalewicz et Schoenauer [Schoenauer & Michalewicz, 1996, 1997] ont proposé une approche qui permet une exploration efficace de la frontière de la région faisable en utilisant des opérateur spéciaux. Cependant, cette méthode ne traite que les problèmes dont l'optimum est sur la frontière.

1.8.3.3 Homomorphous mapping

La méthode *Homomorphous mapping* [Koziel & Michalewicz, 1999] utilise une technique de codage/décodage entre la région faisable $\mathcal{F}$ et le cube unitaire de dimension n :$[-1, 1]^n$. Elle fait ainsi correspondre à chaque individu de la population un individu codé. Cette méthode est difficile à mettre en œuvre et nécessite beaucoup de temps de calcul.

1.8.4 Méthodes hybrides

Les méthodes hybrides ont la caractéristique de séparer la fonction objectif des contraintes. Deux approches présentées dans la littérature permettent de réaliser cette séparation. Nous trouvons d'un côté les méthodes qui traitent les contraintes avec des procédures d'optimisation déterministes, alors que la fonction objectif est toujours optimisée par un EA. Cette approche ne peut pas être appliquée sur des problèmes où la fonction objectif ou une des fonctions des contraintes n'est pas dérivable, puisqu'elle nécessite le calcul du gradient de toutes les fonctions du problème [Waagen et al., 1992]. D'un autre côté, nous distinguons les méthodes dans lesquelles un EA fait lui même la séparation soit en utilisant les techniques multiobjectifs [Parmee & Purchase, 1994; Surry et al., 1995], soit en adoptant le modèle de co-évolution [Paredis, 1994].

1.9 Validation des algorithmes

Comme nous l'avons vu précédemment, il existe de nombreux algorithmes d'optimisation dans des domaines très variés. Une question fondamentale qui se pose ici est : « *Comment pouvons-nous comparer différents algorithmes d'optimisation de manière à pouvoir choisir le plus performant pour un problème d'optimisation donné* ? » Demeurent toutefois des questions relatives à l'utilité d'une métaheuristique particulière pour résoudre un large éventail de problèmes. Selon le *No Free Lunch Theorem*, énoncé en 1997 par D. H. Wolpert et W. G. Macready dans [Wolpert & Macready, 1997], aucun algorithme d'optimisation n'est plus adapté que les autres pour résoudre tous les types de problèmes. Néanmoins, cela n'empêche pas certains algorithmes d'être mieux que d'autres sur des classes particulières de problèmes.

Des protocoles de tests génériques pour la comparaison d'algorithmes existent et nous fournissent des éléments de réponse. En effet, plusieurs fonctions analytiques de tests, regroupées sous l'appellation de « *benchmarks* » , ont été mises en place pour évaluer les performances et les capacités de convergence des algorithmes d'optimisation.

1.9.1 Problèmes de test

Les fonctions proposées par les *benchmarks* possèdent certaines caractéristiques que l'on retrouve dans des problèmes réels difficiles. Elles peuvent être classées selon les propriétés suivantes [Gardeux, 2011] :

- *Unimodalité* : Les *fonctions unimodales* n'ont qu'un seul optimum local, qui est donc l'optimum global. Elles sont, la plupart du temps, considérées comme des fonctions « faciles ». La résolution du problème peut se faire par des techniques d'optimisation locales.
- *Multimodalité* : On considère d'un côté les fonctions « *faiblement multimodales* » qui possèdent peu d'optimums locaux, mais en densité relativement faible par rapport à la taille de l'espace de recherche. Ces fonctions sont plus faciles à résoudre que les fonctions « *hautement multimodales* » qui, elles, possèdent un *grand* nombre d'optimums locaux.
- *Discontinuité* : Une fonction qui présente des « sauts » est dite discontinue. La notion de saut correspond à l'existence d'une limite à droite et d'une limite à gauche qui ne valent pas la même chose.
- *Séparabilité* : Une autre caractéristique importante des fonctions de *benchmarks* est leur séparabilité. On dit qu'une fonction est séparable si l'on peut la minimiser (ou la maximiser) en minimisant (respectivement maximisant) ses composantes prises séparément.
- *Rugosité* : La rugosité donne une indication sur la difficulté à résoudre un problème à l'aide d'une recherche locale utilisant un voisinage donné. En effet, si cette mesure est élevée, on peut supposer que le voisinage est peu adapté car les valeurs des solutions voisines semblent peu corrélées (la fonction objectif représente un *paysage très irrégulier*), alors que si cette mesure est faible, on aura un *paysage lisse*, avec relativement peu d'optima locaux, donc un problème facile à optimiser à l'aide d'une recherche locale [Taillard, 2002].

Plusieurs *benchmarks* existent, dont les plus répandus sont ceux élaborés dans le cadre de conférences. Nous pouvons citer : *Congress on Evolutionary Computation* (CEC), *Black-Box Optimisation Benchmarking* (BBOB). Dans ces *benchmarks*, les optimums globaux sont connus à l'avance. À la fin d'une simulation, il est donc possible d'évaluer l'erreur commise par l'algorithme. Les *benchmarks* sur lesquels nous travaillons sont dédiés à la « *black-box optimization* », c'est-à-dire que nous n'avons aucune information sur la nature de la fonction objectif, nous ne pouvons donc pas calculer sa dérivée.

1.10 Conclusion

Nous avons présenté dans ce chapitre l'état de l'art des métaheuristiques d'optimisation. Après avoir rappelé quelques notions préliminaires, nous avons passé en revue les principaux algorithmes de l'état de l'art, en les divisant en deux classes : les méthodes qui font évoluer une seule solution et les méthodes à base de population de solutions. Un intérêt particulier a été porté à la méthode d'optimisation basée sur la biogéographie (BBO : *Biogeography-based optimization*). Cette jeune méthode, inspirée de la théorie de l'équilibre dynamique, en est encore à ses balbutiements. Elle nous servira de support au long de cette thèse pour présenter les différents travaux que nous avons réalisés.

Nous avons conçu, durant cette thèse, des algorithmes d'optimisation qui se veulent une amélioration de BBO, que nous allons présenter dans les chapitres 2 et 3 suivants. Ces algorithmes sont conçus pour résoudre des problèmes d'optimisation globales à variables continues avec et sans contraintes et seront analysés en utilisant les jeux de tests principalement utilisés dans la littérature.

Chapitre Deux

Contribution au perfectionnement de BBO : Algorithme d'optimisation sans contraintes

2.1 Introduction

Ce chapitre a pour objet de présenter les différentes adaptations que nous avons faites de la métaheuristique classique *Biogeography-based optimization* (BBO) pour les problèmes d'optimisation continue sans contraintes.

Nous avons présenté dans le chapitre 1 une description générale de l'algorithme BBO, qui va nous servir de base pour présenter les différentes adaptations que nous avons réalisées. Dans ce chapitre, nous allons décrire, plus en détail, l'algorithme BBO et en présenter une analyse expérimentale, en vue d'examiner l'influence de ses différents paramètres, sur quelques fonctions de *benchmark* courantes pour les problèmes d'optimisation. Puis dans un second temps, nous présenterons l'adaptation que nous proposons de celui-ci.

Nous avons pu constater, à travers une première analyse de BBO, que la désactivation de l'opérateur de mutation n'a pas de grande influence sur les résultats obtenus. Or, nous savons que la mutation est essentielle car elle assure la diversification de la population, en explorant l'espace des solutions à la recherche du ou des optima globaux. Ce manque de diversité conduit souvent à des solutions sous-optimales. Afin d'améliorer les capacités d'exploration de l'espace de recherche et d'augmenter la diversité de la population, une hybridation avec l'algorithme à évolution différentielle (DE) est proposée. Le choix de DE est motivé par son efficacité et parce qu'il est considéré comme l'un des meilleurs algorithmes pour l'optimisation continue [Noman & Iba, 2008].

L'algorithme proposé, appelé *2-StageBBO-DE* [Boussaïd et al., 2011a], permet, tout en évitant les minima locaux, d'améliorer la qualité de la solution trouvée en raison d'une capacité d'exploration plus grande due à une stratégie de mutation aléatoire et à un croisement qui favorise la diversification. Le test de cette nouvelle méthode, dans le contexte d'un ensemble très large de fonctions continues mono-objectifs, s'est avéré satisfaisant dans le sens où elle permet d'atteindre souvent une meilleure performance, sur plusieurs cas de test de référence, que celle atteinte en utilisant BBO ou DE seuls. Nous avons également introduit des variantes de *2-StageBBO-DE* en choisissant parmi différentes stratégies de mutation de l'algorithme DE. Ces

méthodes ont été aussi analysées et comparées à *2-StageBBO-DE* avec le schéma de mutation sélectionné. Finalement, l'algorithme *2-StageBBO-DE* a été évalué, dans le cadre d'une étude comparative avec d'autres méthodes de la littérature.

2.2 L'algorithme BBO - Principe de fonctionnement

Nous rappelons dans cette section le principe de fonctionnement de BBO. Comme nous l'avons vu dans le chapitre 1, section 1.6.3.4, l'algorithme BBO s'inspire de la théorie de la *bio-géographie insulaire*. Les deux principaux opérateurs qui gouvernent son fonctionnement sont la migration et la mutation. L'algorithme commence par générer un nombre fini d'individus choisis généralement par tirage aléatoire uniforme dans l'espace de recherche formant la population initiale. Après évaluation de la population initiale, certains individus sont choisis pour participer à l'opération de migration qui permet de créer un nouvel ensemble d'individus (à noter que cette étape est stochastique et dépend des taux d'émigration et d'immigration des individus impliqués). Les descendants vont être à leur tour mutés. Le taux de mutation fixe la proportion de la population qui sera renouvelée à chaque génération. L'élitisme permet la conservation du ou des meilleurs individus trouvés, tant qu'ils ne sont pas dépassés par d'autres. Enfin, une phase de remplacement consiste à remplacer les parents par les nouveaux descendants, afin de former une nouvelle population, de la même taille qu'au début de l'itération. L'algorithme effectue un certain nombre de générations, qui se définit le plus souvent en fonction de la taille de la population et du temps imparti pour résoudre le problème.

L'algorithme 1.1 illustre le principe général de fonctionnement de BBO. Nous détaillons ci-après les différentes étapes de l'algorithme.

2.2.1 Génération de la population initiale

La population initiale est générée par tirage aléatoire uniforme sur l'ensemble de valeurs possibles de chaque variable. Les bornes inférieures et supérieures des variables sont spécifiées par l'utilisateur selon la nature du problème.

Pour plus de clarté, *NP* désigne, dans la suite, le nombre d'individus de la population. Pour bien distinguer chaque variable de décision (ou *SIV* :*Suitability Index Variable*, selon la terminologie employée dans BBO) de chaque individu à chaque génération, $X_{i,j,g}$ représente le $j^{\text{ème}}$ *SIV* ($j = 1, \ldots, D$) du $i^{\text{ème}}$ individu ($i = 1, \ldots, NP$) de la $g^{\text{ème}}$ génération. Un individu i de la population, à la génération g, est représentée par :

$$\vec{X}_{i,g} = (X_{i,1,g}, X_{i,2,g}, X_{i,j,g}, \ldots, X_{i,D,g})$$

où D est la dimension de l'espace de recherche. Chaque SIV prend ses valeurs dans un intervalle $[L_j, U_j]$. Le processus d'initialisation génère aléatoirement les NP individus, de façon à couvrir l'espace de recherche de manière uniforme, comme suit :

$$X_{i,j,g} = L_j + rand(0,1) \times (U_j - L_j) \tag{2.2.1}$$

où L_j et U_j sont les bornes inférieures et supérieures de la variable $X_{i,j,g}$ et la fonction $rand$ génère des valeurs aléatoires uniformément dans l'intervalle $[0, 1]$.

2.2.2 Migration

L'idée de la migration est de créer des descendants à partir de plusieurs individus de la population (voir figure 2.1). Le nombre et la destination des SIVs migrants est décidé aléatoirement en fonction des taux d'émigration et d'immigration, comme le montre l'algorithme 2.1.

Algorithme 2.1: Opérateur de migration de l'algorithme BBO

Calculer les taux d'immigration (λ_i) et d'émigration (μ_i) pour chaque habitat $\vec{X}_{i,g}$ ($i = 1, \ldots, NP$)
$\lambda_{\min} = \min(\lambda_i)$ pour $i = 1, \ldots, NP$
$\lambda_{\max} = \max(\lambda_i)$ pour $i = 1, \ldots, NP$
pour $i = 1$ à NP **faire**
    Normaliser le taux d'immigration
    $\lambda_{Scale} = (\lambda_i - \lambda_{\min})/(\lambda_{\max} - \lambda_{\min})$
    pour $j = 1$ à D **faire**
        si ($rand$(0,1) $< \lambda_{Scale}$) **alors**
            $Random_j = rand(0,1) \times \sum_{i=1}^{NP}(\mu_i)$
            $Select = \mu_1$
            $k = 1$
            tant que (($Random_j > Select$) & ($k < NP$)) **faire**
                $k = k + 1$
                $Select = Select + \mu_k$
            fin
            $M_{i,j,g} = X_{k,j,g}$;
        sinon
            $M_{i,j,g} = X_{i,j,g}$
        fin
    fin
fin

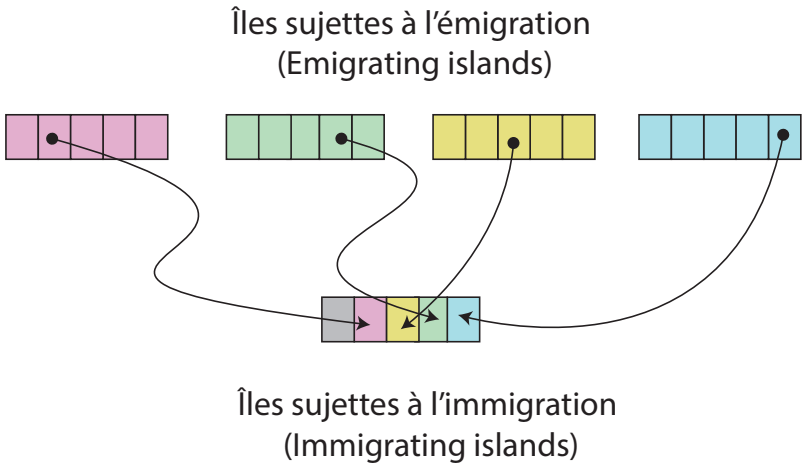


FIGURE 2.1: Le processus de migration dans BBO

2.2.3 Mutation

La mutation, ou perturbation aléatoire des individus, agit comme un opérateur de dispersion au sein de la population, elle permet ainsi de maintenir la diversité. Durant la phase de mutation, un individu i a une probabilité p_m de muter. Dans le cas d'un codage réel, on utilise principalement la mutation uniforme. En supposant fixée la probabilité de mutation p_m, un tirage au sort pour chaque SIV $X_{i,j,g}$ d'un habitat $X_{i,g}$ permet de décider si ce SIV doit être ou non modifié. Nous supposons que le SIV prend ses valeurs dans un intervalle $[L_j, U_j]$. On remplace le SIV $X_{i,j,g}$ sélectionné par une valeur quelconque $X'_{i,j,g}$ tirée aléatoirement dans l'intervalle $[L_j, U_j]$.

$$X'_{i,j,g} = L_j + rand(0,1) \times (U_j - L_j) \tag{2.2.2}$$

2.2.4 Élitisme

La stratégie élitiste consiste à conserver le(s) meilleur(s) individu(s) à chaque génération. Ainsi l'élitisme empêche l'individu le plus performant de disparaître au cours du remplacement ou que ses bonnes combinaisons soient affectées par les opérateurs de variation. Après chaque évaluation de la performance des individus à une génération g donnée, les n_{elit} meilleurs individus de la génération précédente $(g-1)$ sont réintroduits dans la population si aucun des individus de la génération g n'est meilleur qu'eux.

Paramètres	Notation	Valeur
Taille de la population	NP	100
Probabilité de mutation	p_m	0,01
Taille de la mémoire élite	n_{elit}	2
Taux d'immigration maximal	I	1
Taux d'émigration maximal	E	1
Nombre maximum de générations	$g_{\max}$	100

TABLEAU 2.1: Paramètres de l'algorithme BBO

2.3 Étude de l'influence des paramètres

Nous nous proposons dans cette section d'étudier l'influence des différents paramètres qui régissent le fonctionnement de BBO en terme de qualité de solutions trouvées. Pour cela nous allons comparer les résultats obtenus à l'aide de différentes versions de l'algorithme. Nous décidons de tester ici l'influence du nombre d'individus présents dans la population, du taux de mutation et du paramètre d'élitisme. Nous nous limitons à un nombre maximum de génération égal à 100. Les résultats correspondent à la moyenne statistique réalisée sur 25 essais. Les paramètres de BBO sont disponibles dans le tableau 2.1.

2.3.1 Cas de test

Puisque l'on s'intéresse à l'optimisation de fonctions, disons à la minimisation pour fixer les idées, l'objectif est de trouver un vecteur x^* tel que $f(x^*)$ soit un minimum global de la fonction objectif f, défini de $\mathbb{R}^n$ dans $\mathbb{R}$. Autrement dit, nous cherchons x^* tel que $x^* \in arg\min_{x\in\mathbb{R}^n} f(x)$. Pour la maximisation, la correspondance se fait en inversant le critère d'évaluation.

En vue d'examiner l'influence des différents paramètres, nous avons choisi quelques fonctions de *benchmark* courantes pour les problèmes d'optimisation. Une étude expérimentale sera réalisée sur ces cas de test. Nous avons choisie de limiter la dimension de l'espace de recherche à $D = 20$.

2.3.1.1 Problèmes unimodaux

Sphère La fonction *Sphere* [Rechenberg, 1973; DeJong, 1975], est une des fonctions objectifs les plus simples à résoudre, de part sa nature séparable (c'est-à-dire que les variables sont indépendantes – elle peut être réécrite comme somme de fonctions d'une variable) et unimodale. La cavité du minimum est explicite, la représentation est d'allure parabolique. La fonction *Sphere* correspond à l'équation suivante :

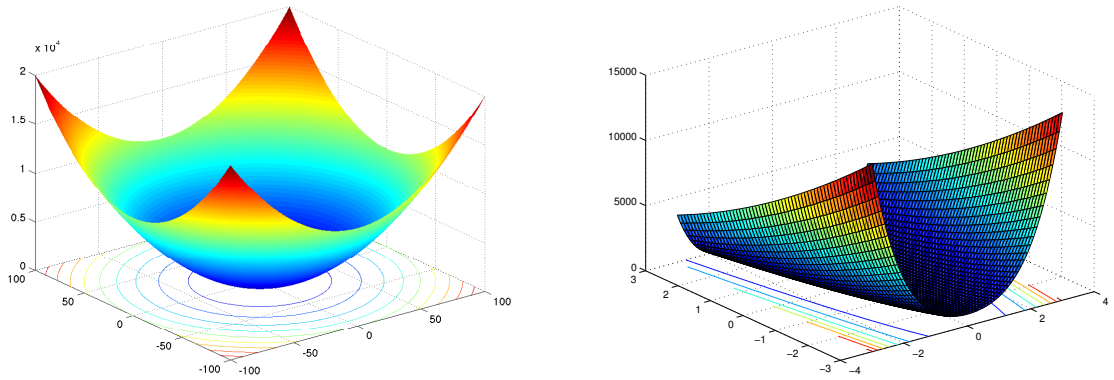
$$F_{Sphere}(x) = \sum_{i=1}^{D} x_i^2 \tag{2.3.1}$$

où D désigne le nombre de dimensions du vecteur solution x. La figure 2.2(a) représente graphiquement la fonction *Sphere*. L'espace de recherche est ici limité à $[-100, 100]$ pour chaque x_i, la troisième composante est donc l'image de x par la fonction objectif ($f(x)$). Le minimum global est obtenu au point $(x_1, x_2) = (0,0)$ (pour $D = 2$), pour lequel la fonction vaut 0.

Rosenbrock La fonction de *Rosenbrock* est une fonction unimodale, comme la fonction *Sphere*, mais non séparable au contraire de cette dernière. Elle a été introduite par Rosenbrock en 1960 [Rosenbrock, 1960]. Elle est aussi connue sous le nom de *fonction banane*. La fonction présente un minimum global à l'intérieur d'une longue vallée étroite de forme parabolique. Trouver la vallée est trivial. Par contre, converger vers le minimum global est difficile. La fonction est définie par :

$$F_{Rosenbrock}(x) = \sum_{i=1}^{D-1} \left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right] \tag{2.3.2}$$

La figure 2.2(b) représente graphiquement la fonction *Rosenbrock*. L'espace de recherche est ici limité à $[-3, 3]$, le minimum global est obtenu au point (1,1), pour lequel la fonction vaut 0.



(a) (b)

FIGURE 2.2: Problèmes unimodaux : (a) fonction *Sphere*, (b) fonction *Rosenbrock*

2.3.1.2 Problèmes multimodaux

Rastrigin La fonction *Rastrigin* est une fonction trigonométrique non-convexe en « champs de bosses », utilisée en tant que problème de test des performances pour des algorithmes d'optimisation. Il s'agit d'un exemple typique de fonction non linéaire multimodal. Il a d'abord été proposé par Rastrigin [Törn & Zilinskas, 1989] en fonction 2-dimensionnelle

et a été généralisé par Mühlenbein et al. [Mühlenbein et al., 1991]. Cette fonction est un problème assez difficile à cause de son grand espace de recherche et de son grand nombre de minima locaux. Elle est définie par :

$$F_{Rastrigin}(x) = \sum_{i=1}^{D} \left(x_i^2 - 10cos(2\pi x_i) + 10\right) \tag{2.3.3}$$

La fonction est représentée graphiquement par la figure 2.3(a). L'espace de recherche est ici limité à $[-5.12, 5.12]$, le minimum global est obtenu au point (0,0), pour lequel la fonction vaut 0.

Ackley La fonction *Ackley* est une des fonctions les plus largement utilisée. Elle possède de nombreux minima locaux, rendant la recherche difficile. Elle est définie par :

$$F_{Ackley}(x) = -20exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - exp\left(\frac{1}{D}\sum_{i=1}^{D} cos\left(2\pi x_i\right)\right) + 20 + e$$

La fonction est représentée graphiquement par la figure 2.3(b). L'espace de recherche est ici limité à $[-10, 10]$, le minimum global est obtenu au point (0,0), pour lequel la fonction vaut 0.

Griewank La fonction *Griewank* [Griewank, 1981] est largement utilisée pour tester la convergence des algorithmes d'optimisation. C'est une fonction hautement multimodale, car le nombre de minima croît de façon exponentielle quand le nombre de dimensions augmente. La fonction est définie comme suit :

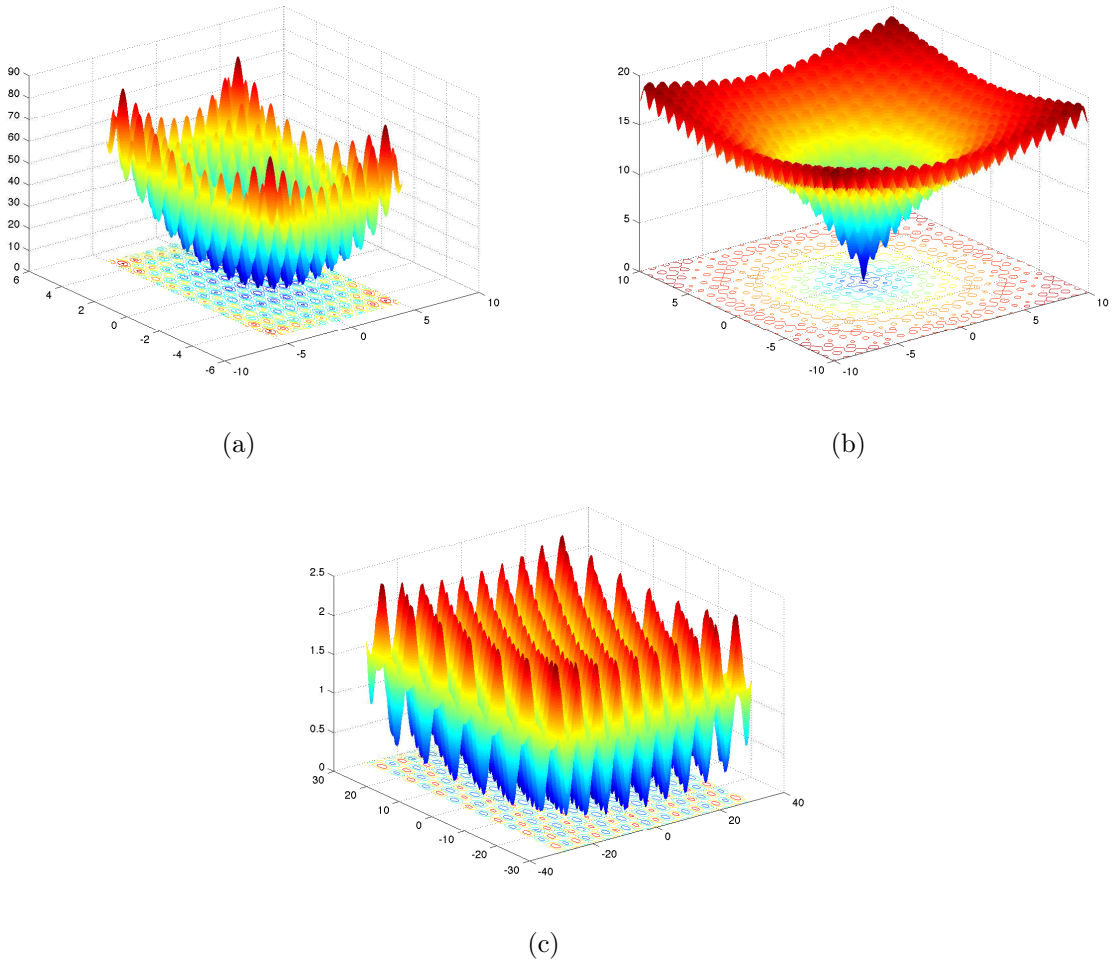
$$F_{Griewank}(x) = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$$

La fonction est représentée graphiquement par la figure 2.3(c). L'espace de recherche est ici limité à $[-600, 600]$, le minimum global est obtenu au point (0,0), pour lequel la fonction vaut 0.

2.3.2 Effets sur l'évolution si on change la taille de la population

Le tableau 2.2 présente les résultats obtenus en moyenne pour 25 exécutions successives de l'algorithme BBO pour les différentes fonctions de test. On constate que la configuration comportant 100 individus semble être la plus favorable. Mais il est inutile d'augmenter ce paramètre de manière démesurée. En effet, plus le nombre d'individus est grand, plus la probabilité de convergence est élevée, mais plus l'algorithme est lent pour calculer chaque génération.

Nous donnons dans la figure 2.4 quelques courbes d'évolution du meilleur individu pour les configurations implémentées.



(a) (b)

(c)

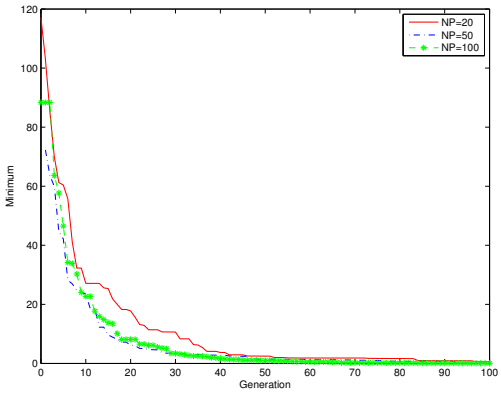
FIGURE 2.3: Problèmes multimodaux : (a) fonction *Rastrigin*, (b) fonction *Ackley*, (c) fonction *Griewank*

	NP = 20	NP = 50	NP = 100
Sphere	1.8202	0.5761	**0.2387**
Rosenbrock	105.4915	73.4287	**46.3181**
Rastrigin	33.2444	17.4333	**9.9805**
Ackley	7.5037	5.0377	**3.8733**
Griewank	7.1870	2.7988	**1.6907**

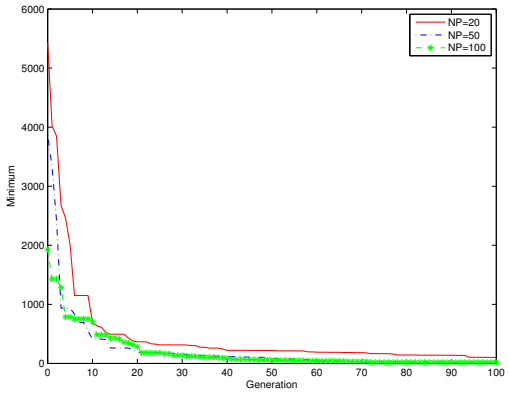
TABLEAU 2.2: Influence de la taille de la population

2.3.3 Effets sur l'évolution si on change le taux de mutation

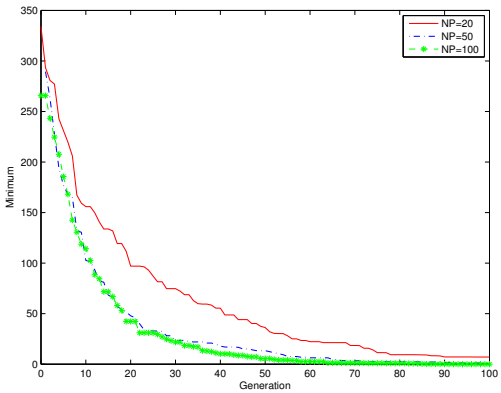
L'algorithme BBO a été testé avec les paramètres du tableau 2.1, en faisant varier le taux de mutation. D'après les résultats de l'expérience, consignés dans le tableau 2.3, il est évident que l'opérateur mutation est un paramètre secondaire dans l'algorithme BBO. En effet, la désactivation de l'opérateur de mutation n'a pas de grande influence sur les résultats. En plus, si on considère le cas avec un taux de 0,01 et celui avec 0 on ne remarque pas de très grande



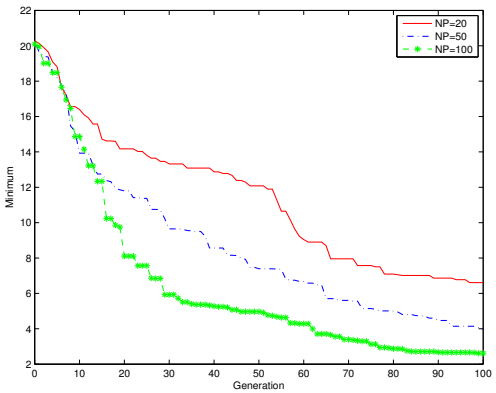
(a)



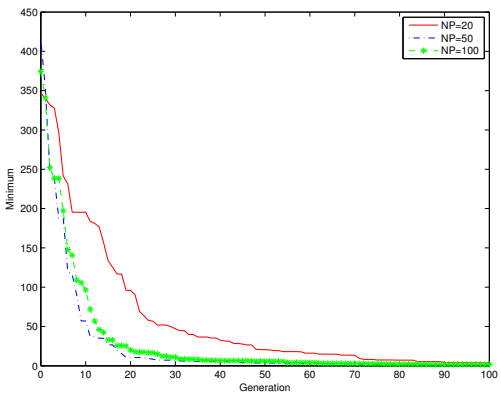
(b)



(c)



(d)



(e)

FIGURE 2.4: Influence de la taille de la population : (a) Sphere, (b) Rosenbrock, (c) Rastrigin, (d) Ackley, (e) Griewank

différence dans les résultats. Pour ce paramètre aussi il est inutile de l'augmenter de manière démesurée : en effet la dernière simulation, avec le taux de mutation de 0,5, montre clairement que les résultats se sont dégradés. On constate, toutefois, que la configuration avec un taux de

mutation de 0,01 semble être la plus favorable.

	$P_m = 0$	$P_m = 0{,}01$	$P_m = 0{,}05$	$P_m = 0{,}1$	$Pm = 0{,}5$
Sphere	0.2508	**0.2387**	1.4634	5.9419	49.0000
Rosenbrock	60.1141	**46.3181**	89.6424	204.6876	2109.9322
Rastrigin	**9.9622**	9.9805	17.3656	49.6529	39.0000
Ackley	4.0437	**3.8733**	6.8123	11.2332	16.6021
Griewank	1.8250	**1.6907**	5.5579	22.6892	139.6625

TABLEAU 2.3: Influence du taux de mutation

La figure 2.5 présente les courbes d'évolution du meilleur individu au cours des 100 générations pour les schémas de mutation implémentés.

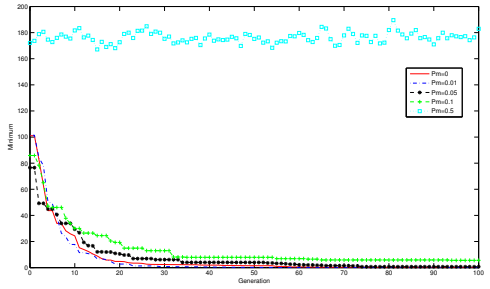
2.3.4 Effets sur l'évolution si on change le paramètre d'élitisme

Pour chaque cas de test, 25 essais ont été effectués dans les mêmes conditions expérimentales que celle décrite dans le tableau 2.1, en faisant varier le paramètre d'élitisme. Ce paramètre correspond au nombre d'individus de la génération précédente que l'on retrouvera dans la prochaine : les individus choisis seront les meilleurs. Les résultats des différentes expériences sont résumés dans le tableau 2.4.

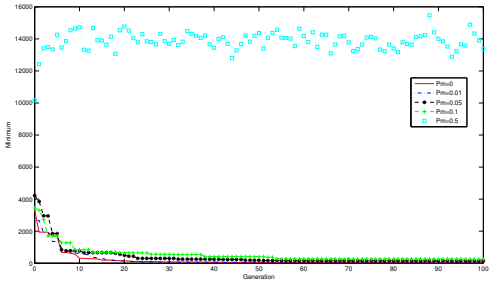
D'après les simulations réalisées, on voit clairement que si on désactive le mécanisme d'élitisme (donc $n_{elit} = 0$), on obtient de moins bons résultats. Les performances de l'algorithme augmentent en incrémentant la valeur de n_{elit}. Par contre, trop augmenter le facteur d'élitisme peut nuire à la diversité de la population. Là aussi, on ne peut pas augmenter de manière démesurée ce paramètre.

	$n_{elit} = 0$	$n_{elit} = 2$	$n_{elit} = 4$	$n_{elit} = 6$
Sphere	0.2979	0.2387	0.2467	**0.2310**
Rosenbrock	54.3107	46.3181	54.4384	**41.3617**
Rastrigin	10.7069	9.9805	9.5225	**9.2541**
Ackley	4.0587	3.8733	3.7019	**3.5830**
Griewank	1.9383	1.6907	1.6476	**1.6312**

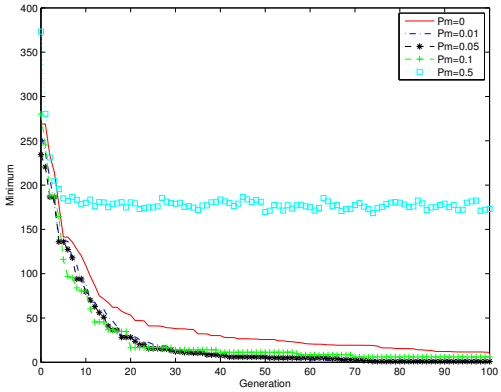
TABLEAU 2.4: Influence de l'élitisme



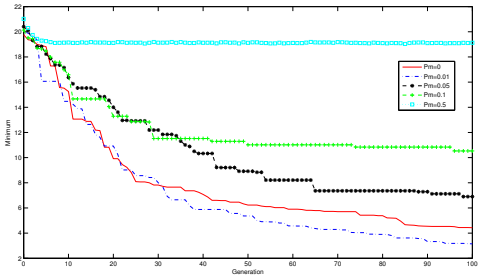
(a)



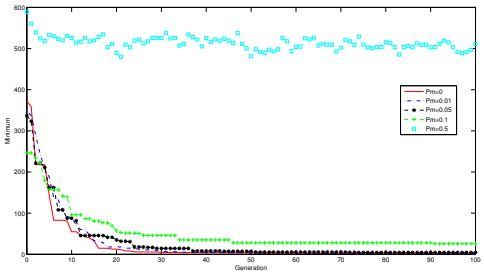
(b)



(c)

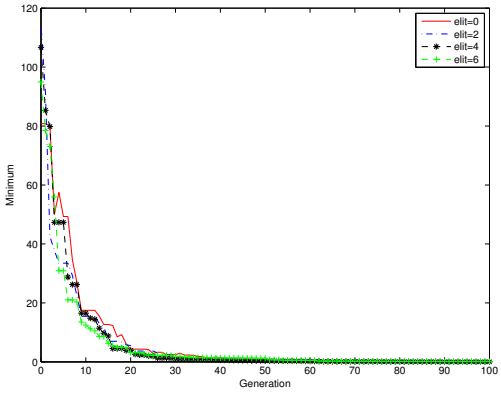


(d)

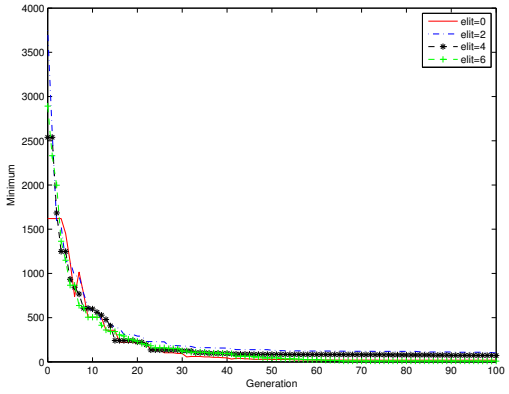


(e)

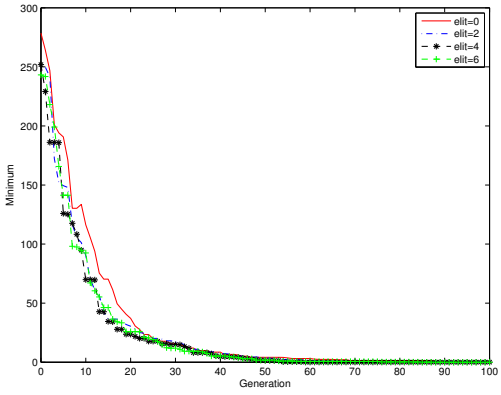
FIGURE 2.5: Influence du taux de mutation : (a) Sphere, (b) Rosenbrock, (c) Rastrigin, (d) Ackley, (e) Griewank



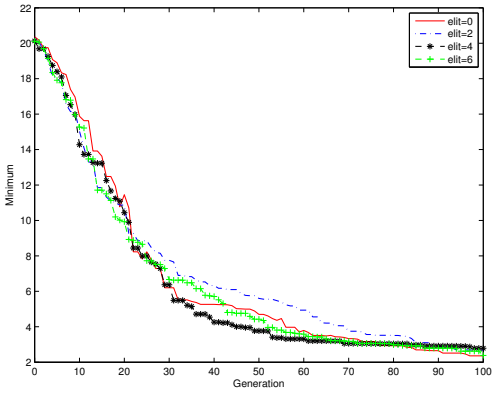
(a)



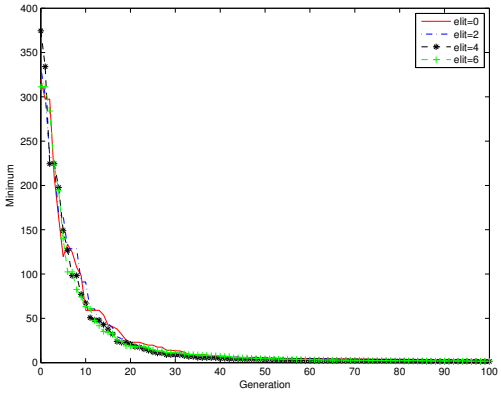
(b)



(c)



(d)



(e)

FIGURE 2.6: Influence du paramètre d'élitisme : (a) Sphere, (b) Rosenbrock, (c) Rastrigin, (d) Ackley, (e) Griewank

2.4 Amélioration de BBO – Méthode *2-StageBBO-DE*

Dans la méthode BBO classique, nous pouvons considérer que la tâche d'intensification est en grande partie remplie par la migration, alors que celle de diversification est exécutée par la mutation. Or, nous avons pu constater, à travers l'analyse conduite dans la section précédente, que la mutation, sensée agir comme un opérateur de dispersion au sein de la population, n'a pas de grande influence sur les résultats. L'opérateur de mutation est essentiel car il assure la diversification de la population, en explorant l'espace des solutions à la recherche du ou des optima globaux. Ainsi, lorsque l'impact de la mutation est faible, il en résulte un manque de diversification, conduisant souvent à des solutions sous-optimales.

Afin de se prémunir au mieux des problèmes de minima locaux, nous avons élaboré une variante autour des algorithmes BBO et DE classiques. Ce nouvel algorithme, baptisé *2-StageBBO-DE*, devrait nous permettre, en évitant les minima locaux, d'améliorer la qualité de la solution trouvée. Nous présentons dans ce qui suit le principe de fonctionnement de l'algorithme proposé.

2.4.1 Principe

À travers la nouvelle méthode proposée, nous avons tenté de combiner, d'un côté, les mouvements dans l'espace de recherche guidés par la technique de recombinaison et de mutation de l'algorithme DE et de l'autre, l'exploitation et la diminution de l'amplitude du mouvement au fur et à mesure que l'on s'approche d'une solution à travers l'utilisation des opérateurs de l'algorithme BBO. L'intérêt de l'approche proposée est de pouvoir bénéficier des avantages de ces méthodes dans une perspective d'amélioration de BBO. Deux stratégies de mise à jour de chaque individu ont été mises en œuvre dans *2-StageBBO-DE* : *Differential Evolution Updating Strategy* et *BBO Updating Strategy*. La première stratégie, est exécutée toutes les k générations (c'est-à-dire lorsque $g\%k = 0$, où g désigne la génération courante et le paramètre k indique la fréquence à laquelle l'algorithme DE est exécuté). Les deux processus, dans notre approche hybride, partagent une seule et même population. L'algorithme commence par l'activation de l'algorithme DE qui met à jour la population, puis transfère la commande à l'algorithme BBO. La stratégie de mise à jour de la population dans *2-StageBBO-DE* est décrite ci-après. Le pseudo-code présenté dans l'algorithme 2.2 décrit la méthode développée.

Algorithme 2.2: L'algorithme *2-StageBBO-DE*

Entrées : $g_{\max}$: nombre maximal de générations, f : fonction objectif, p_m : probabilité de mutation, D : nombre de SIVs, NP : nombre d'individus, n_{elit} : nombre d'individus conservés entre les générations, E : taux d'émigration maximal, I : taux d'immigration maximal, $S_{\max}$: nombre maximum d'espèces par habitat, F : constante de différentiation, CR : taux de croisement.

Sorties : $\vec{X}_{opt}$ minimisant f

Génération de la population initiale par tirage aléatoire uniforme sur l'ensemble des valeurs possibles de chaque variable
Évaluation de la population initiale
Initialisation du compteur de générations $g = 0$
tant que $g < g_{\max}$ **faire**
    si $g\%k = 0$ **alors**
        pour $i = 1$ à NP **faire**
            Mettre à jour la solution actuelle $\vec{X}_i$ suivant la stratégie DE (cf. section 2.4.2.2)
        fin
        Évaluation de la nouvelle population
        Sélection des meilleurs individus pour la prochaine génération
    sinon
        Tri des individus par ordre décroissant de *fitness*
        Sélection des n_{elit} meilleurs individus et constitution d'une mémoire élite
        pour $i = 1$ to NP **faire**
            Calculer le nombre d'espèces par habitat : $SpeciesCount_i = S_{\max} - i$
            Calculer les taux d'immigration λ_i et d'émigration μ_i pour chaque individu $\vec{X}_i$:
            $\mu_i = E(SpeciesCount_i / S_{\max})$
            $\lambda_i = 1 - \mu_i$
            Mettre à jour $\vec{X}_i$ suivant la stratégie BBO (cf. section 2.4.2.1)
        fin
        Évaluation de la nouvelle population
        Sélection des meilleurs individus pour la prochaine génération
        Intégration des individus de la mémoire élite et remplacement des individus les plus mauvais
    fin
    $g = g + 1$
fin
retourner la meilleure solution trouvée, $\vec{X}_{opt}$

2.4.2 Stratégie de mise à jour de la population

Dans l'algorithme *2-StageBBO-DE*, la population est mise à jour en appliquant alternativement les stratégies de mise à jour des algorithmes BBO et DE, avec la stratégie DE, exécutée toutes les k générations.

2.4.2.1 *BBO updating strategy*

Les étapes de mise à jour de la population sont décrites dans la section 2.2. Les individus sont soumis aux opérateurs de variation, à savoir la migration et la mutation. Contrairement à l'algorithme BBO classique où la nouvelle génération d'individus remplace l'ancienne, *2-StageBBO-DE* utilise un opérateur de sélection dans le but de garder les individus les mieux adaptés, i.e. les individus correspondant aux valeurs de *fitness* les plus faibles, pour la prochaine génération. La règle de sélection est la suivante :

$$\vec{X}_{i,g} = \begin{cases} \vec{M}_{i,g} & \text{si } f(\vec{M}_{i,g}) < f(\vec{X}_{i,g}) \\ \vec{X}_{i,g} & \text{si } f(\vec{M}_{i}) > f(\vec{X}_{i}) \end{cases} \tag{2.4.1}$$

2.4.2.2 *DE updating strategy*

L'algorithme à évolution différentielle (DE) a été introduit au Chapitre 1 (section 1.6.2.2). Il fait évoluer les individus de la population par mutation et croisement des individus. Chaque individu $\vec{X}_{i,g}$ de la population est caractérisé par un vecteur $(X_{i,1,g}, \ldots, X_{i,D,g})^T$ à la génération g, avec D la dimension de l'espace de recherche. Différentes forme de mutation existent, dont la notation est la suivante : *DE/x/y/z*. La variable x fait référence au mode de sélection (aléatoire ou non) du vecteur de référence pour la mutation, la variable y détermine le nombre de différentiations utilisées lors de la mutation et la variable z représente le mode de croisement. Nous présentons ci-après les schémas de mutation les plus couramment utilisés (cf. figure 2.7) :

- **DE/rand/1** : Pour chaque vecteur $\vec{X}_{i,g}$ de la génération g, on construit le vecteur mutant $\vec{V}_{i,g}$ à partir de trois vecteurs $\vec{X}_{r_1,g}$, $\vec{X}_{r_2,g}$ et $\vec{X}_{r_3,g}$ aléatoirement choisis dans le reste de la population, tous différents et différents de $\vec{X}_{i,g}$. Le facteur F contrôle l'amplitude du vecteur d'exploration $(\vec{X}_{r_2,g} - \vec{X}_{r_3,g})$:

$$\vec{V}_{i,g} = \vec{X}_{r_1,g} + F(\vec{X}_{r_2,g} - \vec{X}_{r_3,g}) \tag{2.4.2}$$

- **DE/rand/2** : Pour créer le vecteur mutant $\vec{V}_{i,G}$, pour chaque vecteur $X_{i,g}$, un total de cinq autres vecteurs est aléatoirement choisi dans le reste de la population, mutuellement différents et différents de $X_{i,g}$.

$$\vec{V}_{i,g} = \vec{X}_{r_1,g} + F(\vec{X}_{r_2,g} - \vec{X}_{r_3,g}) + F(\vec{X}_{r_4,g} - \vec{X}_{r_5,g}) \tag{2.4.3}$$

– **DE/best/1** : La création du nouvel individu, $\vec{V}_{i,G}$, est réalisée en ajoutant une perturbation au meilleur individu de la population, à travers deux autres individus choisis aléatoirement.

$$\vec{V}_{i,g} = \vec{X}_{best,g} + F(\vec{X}_{r_1,g} - \vec{X}_{r_2,g}) \tag{2.4.4}$$

– **DE/best/2** : Dans ce schéma de mutation, le vecteur mutant est créé en ajoutant une perturbation au meilleur individu à travers deux différences pondérées d'individus sélectionnés aléatoirement dans le reste de la population.

$$\vec{V}_{i,g} = \vec{X}_{best,g} + F(\vec{X}_{r_1,g} - \vec{X}_{r_2,g}) + F(\vec{X}_{r_3,G} - \vec{X}_{r_4,G}) \tag{2.4.5}$$

– **DE/current to best/1** : le vecteur mutant est créé à l'aide de deux vecteurs choisis au hasard, ainsi que le meilleur vecteur de la génération courante.

$$\vec{V}_{i,g} = \vec{X}_{i,g} + F(\vec{X}_{best,g} - \vec{X}_{i,g}) + F(\vec{X}_{r_1,g} - \vec{X}_{r_2,g}) \tag{2.4.6}$$

Les indices r_1, r_2, r_3, r_4, r_5 sont générés par tirage aléatoire uniforme dans l'intervalle $[1, NP]$ et doivent être mutuellement différents et différents de l'indice courant i ; $F \in [0, 1]$ est appelé poids différentiel ; $\vec{X}_{best,g}$ est la meilleure solution trouvée à la génération g.
Après la mutation, une opération de croisement binaire forme le vecteur d'essai final $\vec{U}_{i,g}$ selon le vecteur $\vec{X}_{i,g}$ de la population à la génération g et le vecteur mutant correspondant $\vec{V}_{i,g}$:

$$U_{i,j,g} = \begin{cases} V_{i,j,g} & \text{si } rand(0,1) \leq CR \text{ ou } j = j_{rand} \\ X_{i,j,g} & \text{sinon} \end{cases} \tag{2.4.7}$$

Le taux de croisement CR détermine la distance séparant le vecteur d'essai engendré $\vec{U}_{i,g}$ du vecteur de référence $\vec{X}_{i,g}$. Avec un faible taux de croisement, proche de zéro, la plupart des composantes de $\vec{U}_{i,g}$ sont identiques à celles du vecteur de référence. Si au contraire le taux de croisement est proche de 1, le vecteur d'essai $\vec{U}_{i,g}$ sera très similaire au vecteur mutant $\vec{V}_{i,g}$ qui, selon le schéma de mutation sélectionné peut être situé loin du vecteur de référence, permettant ainsi un plus large rayon d'exploration de l'espace de recherche.

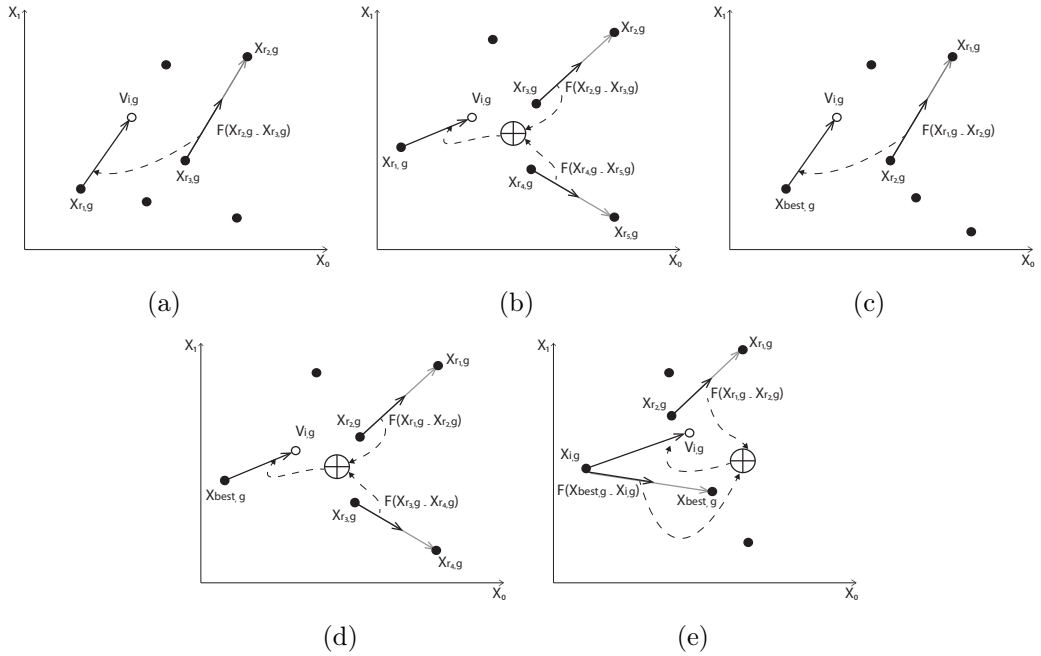


FIGURE 2.7: Exemples de croisement : (a) DE/rand/1, (b) DE/rand/2, (c) DE/best/1, (d) DE/best/2, (e) DE/current to best/1

2.5 Résultats expérimentaux et analyse

2.5.1 Problèmes de test

Afin d'évaluer la pertinence de la nouvelle variante hybride proposée, nous l'avons comparée aux deux algorithmes de base, BBO et DE sur un ensemble de fonctions-tests (*benchmark*). Les fonctions ($F1$ à $F14$) sont des fonctions de *benchmark* de base pour les problèmes d'optimisation [Yao et al., 1999]. Les fonctions $F1$ à $F7$ ne possèdent qu'un minimum global, tandis que les fonctions $F8$ à $F14$ sont des fonctions hautement multimodales, i.e. le nombre de pics et de vallées augmente avec l'augmentation du nombre de dimensions. Les fonctions ($F15$ à $F31$), proposées par la conférence *IEEE Congress on Evolutionary Computation* (CEC 2005) [Liang et al., 2005], sont des versions biaisées et tournées des fonctions de base. Les fonctions $F29$ à $F31$ sont des fonctions composées. Il est à noter que nous n'avons pas gardé les mêmes numéros de fonctions assignés dans le *benchmark* CEC 2005. Les différentes caractéristiques de ces fonctions peuvent être vues dans le tableau 2.5 .

Pour chacune de ces fonctions, le but est de trouver le minimum global $\vec{X}^*$ de F sur $\mathcal{S}$ telle que $F(\vec{X}^*) \leq F(\vec{X}), \forall \vec{X} \in \mathcal{S}\,; \mathcal{S} \subseteq \mathbb{R}^D$ est un ensemble non vide et $\vec{X} \in \mathcal{S}$ est le vecteur solution des n variables de décision $\vec{X} = [X_1, X_2, \ldots, X_n]^D$ où chaque X_i $(i = 1, \ldots, n)$ est bornée par les limites inférieure et supérieure $L_i \leq X_i \leq U_i$ qui définissent l'espace de recherche $\mathcal{S}$; D représente le nombre de dimensions de la fonction et $F(\vec{X}^*)$ désigne la valeur de la fonction à

Fonctions-test		$\mathcal{S}$	D
Fonctions unimodales standards (7)			
F1	Sphere Function	$[-100, 100]^D$	30
F2	Schwefel's 2.22 Function	$[-10, 10]^D$	30
F3	Schwefel's 1.2 Function	$[-100, 100]^D$	30
F4	Schwefel's 2.21 Function	$[-100, 100]^D$	30
F5	Generalized Rosenbrock's Function	$[-30, 30]^D$	30
F6	Step Function	$[-100, 100]^D$	30
F7	Quartic Function with Noise	$[-1.28, 1.28]^D$	30
Fonctions multimodales standards (7)			
F8	Schwefel's 2.26 Function	$[-500, 500]^D$	30
F9	Generalized Rastrigin's Function	$[-5.12, 5.12]^D$	30
F10	Ackley Function	$[-32, 32]^D$	30
F11	Generalized Griewank Function	$[-600, 600]^D$	30
F12	Generalized Penalized Function 1	$[-50, 50]^D$	30
F13	Generalized Penalized Function 2	$[-50, 50]^D$	30
F14	Fletcher Powell Function	$[-\pi, \pi]^D$	30
Fonctions unimodales CEC 2005 (5)			
F15	Shifted Sphere Function	$[-100, 100]^D$	10
F16	Shifted Schwefel's Problem 1.2	$[-100, 100]^D$	10
F17	Shifted Rotated High Conditioned Elliptic Function	$[-100, 100]^D$	10
F18	Shifted Schwefel's Problem 1.2 with Noise in Fitness	$[-100, 100]^D$	10
F19	Schwefel's Problem 2.6 with Global Optimum on Bounds	$[-100, 100]^D$	10
Fonctions multimodales CEC 2005 (12)			
Fonctions basiques(7)			
F20	Shifted Rosenbrock's Function	$[-100, 100]^D$	10
F21	Shifted Rotated Griewank's Function without Bounds	$[0, 600]^D$	10
F22	Shifted Rotated Ackley's Function with Global Optimum on Bounds	$[-32, 32]^D$	10
F23	Shifted Rastrigin's Function	$[-5, 5]^D$	10
F24	Shifted Rotated Rastrigin's Function	$[-5, 5]^D$	10
F25	Shifted Rotated Weierstrass Function	$[-0{,}5,\ 0{,}5]^D$	10
F26	Schwefel's Problem 2.13	$[-\pi,\ \pi]^D$	10
Fonctions étendues (*Expanded functions*) (2)			
F27	Expanded Extended Griewank's plus Rosenbrock's Function	$[-3, 1]^D$	10
F28	Shifted Rotated Expanded Scaffer's	$[-100, 100]^D$	10
Fonctions composées hybrides (*Hybrid composition functions)* (3)			
F29	Hybrid Composition Function	$[-5, 5]^D$	10
F30	Rotated Hybrid Composition Function	$[-5, 5]^D$	10
F31	Rotated Hybrid Composition Function with Noise in Fitness	$[-5, 5]^D$	10

TABLEAU 2.5: Caractéristiques des fonctions du *benchmark* utilisées dans notre étude expérimentale - $\mathcal{S}$ correspond au domaine de définition de chaque fonction et D est le nombre de dimensions.

Paramètres	**Notation**	**Valeur**
Taille de la population	NP	100
Probabilité de mutation	p_m	0,01
Taille de la mémoire élite	n_{elit}	2
Taux d'immigration maximal	I	1
Taux d'émigration maximal	E	1
Fréquence d'exécution de DE	k	2
Constante de differentiation	F	0,5
Taux de croisement	CR	0,9
Stratégie de mutation	*DE/Rand/1/bin*	

TABLEAU 2.6: Paramètres de l'algorithme *2-StageBBO-DE*

l'optimum global.

2.5.2 Paramétrage

Nous présentons dans cette sous-section la liste des paramètres utilisés dans notre étude expérimentale. Les paramètres de l'algorithme *2-StageBBO-DE* sont illustrés dans le tableau 2.6. L'algorithme BBO est utilisé dans sa configuration de base proposée par son auteur [Simon, 2008]. La méthode DE, procède exactement comme l'algorithme original présenté dans [Price et al., 2005]. L'algorithme DE a été expliqué plus en détail dans la section 2.4.2.2. Le schéma de mutation utilisé ici est *DE/rand/1/bin*. Le facteur d'amplification F et la constante de croisement CR ont été fixés respectivement à 0,5 et 0,9 [Storn & Price, 1997]. Pour tous les algorithmes, la taille de la population est fixée à 100.

Des séries d'expérience de 25 tests chacune ont été effectuées pour les fonctions de *benchmark*, présentées dans le tableau 2.5. Le critère d'arrêt des algorithmes est défini en fonction du nombre maximal d'évaluations (N_{Eval}) de la fonction objectif. Ce paramètre est fixé à 100 000 évaluations soit 1 000 générations pour les fonctions issues du *benchmark* CEC 2005. Le nombre d'évaluations, pour les fonctions $F1$ à $F14$, a été suggéré dans [Yao et al., 1999] (voir tableau 2.7).

2.5.3 Résultats et discussions

Les résultats des simulations présentés dans le tableau 2.8 correspondent aux valeurs moyennes (*Mean*) obtenues sur 25 exécutions pour chacun des algorithmes testés et l'écart type (*std.*) entre les différentes exécutions. Les trois algorithmes ont été initialisés de la même manière de sorte que la comparaison soit la plus pertinente possible. La première génération est distribuée uniformément au sein des intervalles de définition de chaque fonction de test (cf. tableau 2.5). Pour plus de clarté, les solutions obtenues, qui sont de meilleure qualité, ont été mises en gras.

Pour les fonctions unimodales ($F1$ à $F7$) l'algorithme *2-StageBBO-DE* obtient de meilleurs

Fonction		$\mathbf{N_{Eval}}$
F1	Sphere Function	150 000
F2	Schwefel's 2.22 Function	200 000
F3	Schwefel's 1.2 Function	500 000
F4	Schwefel's 2.21 Function	500 000
F5	Generalized Rosenbrock's Function	2 000 000
F6	Step Function	150 000
F7	Quartic Function with Noise	300 000
F8	Schwefel's 2.26 Function	900 000
F9	Generalized Rastrigin's Function	500 000
F10	Ackley Function	150 000
F11	Generalized Griewank Function	200 000
F12	Generalized Penalized Function 1	150 000
F13	Generalized Penalized Function 2	150 000
F14	Fletcher Powell Function	500 000
F15 à F31	CEC 2005 Functions	100 000

TABLEAU 2.7: Nombre d'évaluations maximal des fonctions de *benchmark*

résultats qu'avec le BBO classique, pour 6 sur les 7 fonctions. *2-StageBBO-DE* présente des résultats qui sont bien meilleurs que ceux de DE classique pour les fonctions *Sphere function* ($F1$), *Schwefel's Problem 1.21* ($F4$), *Quartic function* ($F7$) et *Step function* ($F6$). On notera cependant que DE montre malgré tout des performances améliorées par rapport au *2-StageBBO-DE* pour les fonctions *Schwefel's 2.22* ($F2$), *Schwefel's 1.2* ($F3$) et *Rosenbrock* ($F5$). Contrairement au DE classique, *2-StageBBO-DE* atteint le minimum global pour la fonction *Step* ($F6$).

Quant aux fonctions multimodales ($F8$ à $F14$) comportant plusieurs minima locaux, nous pouvons voir que notre algorithme obtient de meilleurs résultats que DE classique pour toutes les fonctions sauf pour *Ackley*. Pour les fonctions *Schwefel's 2.26* et *Fletcher Powell*, les résultats obtenus par BBO classique sont meilleurs, comparés aux deux autres algorithmes.

Pour les 17 fonctions du *benchmark* CEC 2005 ($F15$ à $F31$), on peut voir que *2-StageBBO-DE* se détache en obtenant les meilleurs résultats pour 11 fonctions, dont 3 résolues parfaitement (optimum global atteint). DE classique réalise de meilleurs résultats pour 8 fonctions et BBO classique pour 1 fonction. Ces résultats valident davantage la supériorité globale de l'algorithme *2-StageBBO-DE*.

Pour les fonctions ne comportant qu'un minimum global ($F15$ à $F19$), DE obtient des meilleurs résultats que *2-StageBBO-DE* pour les fonctions *Shifted Rotated High Conditioned Elliptic Function* ($F17$) et *the Schwefel's Problem 2.6 with Global Optimum on Bounds* ($F19$), sans pour autant trouver l'optimum global. Pour les fonctions $F15$, $F16$ et $F18$, DE classique et *2-StageBBO-DE* convergent tous les deux vers la solution optimale.

Pour les fonctions multimodales biaisées et tournées et les fonctions composées ($F20$ à $F31$), les résultats obtenus montrent encore une fois la supériorité de notre algorithme. En termes de valeurs moyennes de la fonction objectif, BBO obtient un résultat légèrement supérieur pour

Fonction	DE		BBO		*2-StageBBO-DE*	
	Mean	Std	Mean	Std	Mean	Std
F1	$1{,}10E-22$	$7{,}03E-23$	$4{,}00E-01$	$1{,}55E-01$	$\mathbf{7{,}76E-36}$	$\mathbf{3{,}80E-35}$
F2	$\mathbf{3{,}39E-16}$	$\mathbf{1{,}62E-16}$	$1{,}91E-01$	$4{,}39E-02$	$6{,}14E-04$	$2{,}64E-03$
F3	$\mathbf{2{,}36E-16}$	$\mathbf{4{,}26E-16}$	$5{,}95E+02$	$2{,}77E+02$	$1{,}52E-11$	$1{,}86E-11$
F4	$2{,}00E+00$	$1{,}25E+00$	$9{,}35E-01$	$1{,}69E-01$	$\mathbf{9{,}02E-02}$	$\mathbf{6{,}10E-02}$
F5	$\mathbf{1{,}59E-01}$	$\mathbf{7{,}81E-01}$	$6{,}53E+01$	$3{,}81E+01$	$6{,}71E+01$	$2{,}61E+01$
F6	$1{,}29E-22$	$8{,}03E-23$	$4{,}76E-01$	$2{,}15E-01$	$\mathbf{0{,}00E+00}$	$\mathbf{0{,}00E+00}$
F7	$2{,}37E-03$	$6{,}19E-04$	$6{,}10E-02$	$2{,}17E-02$	$\mathbf{1{,}54E-03}$	$\mathbf{3{,}41E-04}$
F8	$3{,}79E+04$	$1{,}86E+04$	$\mathbf{2{,}79E-02}$	$\mathbf{7{,}86E-03}$	$2{,}10E-01$	$2{,}91E-01$
F9	$1{,}46E+01$	$3{,}97E+00$	$1{,}76E-02$	$5{,}86E-03$	$\mathbf{2{,}24E-03}$	$\mathbf{3{,}25E-03}$
F10	$\mathbf{2{,}62E-12}$	$\mathbf{1{,}09E-12}$	$2{,}11E-01$	$6{,}19E-02$	$4{,}63E-03$	$1{,}47E-02$
F11	$6{,}90E-04$	$2{,}37E-03$	$3{,}05E-01$	$8{,}81E-02$	$\mathbf{1{,}33E-17}$	$\mathbf{6{,}53E-17}$
F12	$5{,}79E-04$	$6{,}00E-04$	$1{,}17E-02$	$1{,}94E-02$	$\mathbf{9{,}64E-06}$	$\mathbf{3{,}68E-05}$
F13	$1{,}39E-02$	$2{,}67E-02$	$2{,}58E-02$	$1{,}03E-02$	$\mathbf{4{,}51E-04}$	$\mathbf{2{,}15E-03}$
F14	$1{,}11E+06$	$1{,}36E+05$	$\mathbf{1{,}08E+06}$	$\mathbf{1{,}09E+05}$	$1{,}10E+06$	$1{,}14E+05$
F15	$\mathbf{0{,}00E+00}$	$\mathbf{0{,}00E+00}$	$1{,}64E-02$	$9{,}87E-03$	$\mathbf{0{,}00E+00}$	$\mathbf{0{,}00E+00}$
F16	$\mathbf{0{,}00E+00}$	$\mathbf{0{,}00E+00}$	$3{,}90E+01$	$3{,}42E+01$	$\mathbf{0{,}00E+00}$	$\mathbf{0{,}00E+00}$
F17	$\mathbf{0{,}00E+00}$	$\mathbf{0{,}00E+00}$	$2{,}00E+06$	$1{,}98E+06$	$1{,}14E-10$	$1{,}28E-10$
F18	$\mathbf{0{,}00E+00}$	$\mathbf{0{,}00E+00}$	$3{,}02E+02$	$2{,}30E+02$	$\mathbf{0{,}00E+00}$	$\mathbf{0{,}00E+00}$
F19	$\mathbf{2{,}62E-12}$	$\mathbf{1{,}16E-12}$	$6{,}67E+02$	$5{,}17E+02$	$1{,}41E-05$	$9{,}16E-06$
F20	$\mathbf{1{,}59E-01}$	$\mathbf{7{,}81E-01}$	$8{,}13E+01$	$4{,}32E+01$	$4{,}14E+00$	$2{,}78E+00$
F21	$\mathbf{5{,}06E-02}$	$\mathbf{3{,}89E-02}$	$1{,}27E+03$	$3{,}85E-01$	$6{,}42E-02$	$8{,}79E-02$
F22	$2{,}04E+01$	$9{,}09E-02$	$2{,}03E+01$	$7{,}91E-02$	$\mathbf{2{,}03E+01}$	$\mathbf{5{,}15E-02}$
F23	$4{,}83E-01$	$6{,}39E-01$	$1{,}04E-02$	$4{,}98E-03$	$\mathbf{4{,}24E-03}$	$\mathbf{1{,}54E-02}$
F24	$7{,}64E+00$	$4{,}56E+00$	$1{,}47E+01$	$5{,}96E+00$	$\mathbf{6{,}93E+00}$	$\mathbf{3{,}28E+00}$
F25	$5{,}49E+00$	$2{,}01E+00$	$5{,}81E+00$	$1{,}65E+00$	$\mathbf{1{,}08E+00}$	$\mathbf{1{,}36E+00}$
F26	$1{,}60E+02$	$3{,}38E+02$	$8{,}53E+02$	$1{,}03E+03$	$\mathbf{1{,}31E+02}$	$\mathbf{4{,}21E+02}$
F27	$6{,}98E-01$	$2{,}88E-01$	$\mathbf{2{,}85E-01}$	$\mathbf{1{,}16E-01}$	$3{,}62E-01$	$9{,}43E-02$
F28	$3{,}52E+00$	$3{,}14E-01$	$3{,}40E+00$	$4{,}55E-01$	$\mathbf{2{,}13E+00}$	$\mathbf{4{,}68E-01}$
F29	$2{,}77E+02$	$1{,}03E+02$	$1{,}71E+02$	$2{,}01E+02$	$\mathbf{6{,}40E+01}$	$\mathbf{1{,}47E+02}$
F30	$1{,}07E+02$	$9{,}03E+00$	$1{,}23E+02$	$1{,}46E+01$	$\mathbf{1{,}02E+02}$	$\mathbf{7{,}31E+00}$
F31	$\mathbf{8{,}43E+01}$	$\mathbf{8{,}30E+00}$	$1{,}32E+02$	$1{,}53E+01$	$9{,}90E+01$	$7{,}40E+00$

TABLEAU 2.8: Comparaison des versions standards de BBO et DE et de l'algorithme *2-StageBBO-DE* sur le *benchmark* des 31 fonctions, donnant pour chaque fonction la performance moyenne et l'écart type sur les 25 essais. Les résultats en gras indiquent le meilleur résultat ou l'optimum global.

la fonction $F27$. Pour les 11 autres fonctions, *2-StageBBO-DE* obtient les meilleurs résultats pour 8 fonctions et DE pour 3 fonctions.

Pour conclure, sur le *benchmark* de 31 fonctions, *2-StageBBO-DE* occupe la première place et se montre particulièrement performant pour les fonctions multimodales. Il obtient les meilleurs résultats dans 19 cas, DE classique dans 12 cas et BBO classique dans 3 cas.

2.5.3.1 Analyse statistique

Pour chaque fonction du *benchmark*, nous avons voulu vérifier si les différences entre les solutions trouvées par *2-StageBBO-DE* et les algorithmes standards BBO et DE (cf. tableau 2.8) étaient statistiquement significatives. À cet effet, nous avons réalisé un test de Student bilatéral.

Formellement, on définit la statistique t par la formule :

$$t = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\frac{S_1^2}{N_1} + \frac{S_2^2}{N_2}}} \tag{2.5.1}$$

où $\bar{X}_1$ et S_1^2 sont respectivement la moyenne et la variance obtenue à partir des résultats de l'algorithme 1 et $\bar{X}_2$ et S_2^2 sont respectivement la moyenne et variance obtenue à partir des résultats de l'algorithme concurrent 2, N_1 et N_2 sont les nombres d'essais réalisés par chacun des algorithmes 1 et 2, respectivement.

La valeur calculée de t doit être confrontée à la table de la loi de Student afin de pouvoir effectivement rejeter ou accepter empiriquement l'hypothèse nulle d'égalité des moyennes. L'hypothèse nulle est rejetée au profit de l'hypothèse alternative, au risque d'erreur α, si la valeur absolue de la statistique t égale ou dépasse une certaine valeur critique, définie en fonction du seuil de probabilité α et du nombre de degrés de liberté $\nu = (N_1 + N_2 - 2)$.

Le tableau 2.9 montre les résultats de la comparaison statistique entre l'algorithme *2-StageBBO-DE* et les deux autres algorithmes, en utilisant le test t avec un nombre de degrés de liberté égal à $\nu = 25 + 25 - 2 = 48$ et un seuil de signification $\alpha = 0{,}05$ (i.e., un degré de confiance de 95%). Nous pouvons résumer les résultats comme suit :

1. ***2-StageBBO-DE* vs. DE** : La valeur calculée de la statistique t dépasse, en valeur absolue, le seuil critique pour 20 fonctions-test sur un total de 31. Ceci signifie que la distance qui sépare les moyennes comparées est trop élevée pour être simplement due au hasard. Pour ces 20 fonctions, nous pouvons rejeter l'hypothèse nulle et en conclure, avec un degré de confiance de 95%, que les moyennes obtenues par les deux algorithmes sont significativement différentes d'un point de vue statistique.
2. ***2-StageBBO-DE* vs. BBO** : Le test de Student conduit à une statistique t supérieure à la valeur critique dans 29 cas. Pour ces fonctions, nous pouvons rejeter l'hypothèse

Fonction	*2-StageBBO-DE* vs. DE	*2-StageBBO-DE* vs. BBO
	t-test	t-test
F1	$-7{,}82E+00\dagger$	$-1{,}29E+01\dagger$
F2	$1{,}16E+00$	$-2{,}17E+01\dagger$
F3	$4{,}09E+00\dagger$	$-1{,}07E+01\dagger$
F4	$-7{,}64E+00\dagger$	$-2{,}35E+01\dagger$
F5	$1{,}28E+01\dagger$	$1{,}99E-01$
F6	$-8{,}02E+00\dagger$	$-1{,}11E+01\dagger$
F7	$-5{,}92E+00\dagger$	$-1{,}37E+01\dagger$
F8	$-1{,}02E+01\dagger$	$3{,}13E+00\dagger$
F9	$-1{,}84E+01\dagger$	$-1{,}14E+01\dagger$
F10	$1{,}58E+00$	$-1{,}62E+01\dagger$
F11	$-1{,}46E+00$	$-1{,}73E+01\dagger$
F12	$-4{,}73E+00\dagger$	$-3{,}02E+00\dagger$
F13	$-2{,}51E+00\dagger$	$-1{,}21E+01\dagger$
F14	$-1{,}93E-01$	$8{,}52E-01$
F15	$0{,}00E+00$	$-8{,}30E+00\dagger$
F16	$0{,}00E+00$	$-5{,}69E+00\dagger$
F17	$4{,}45E+00\dagger$	$-5{,}04E+00\dagger$
F18	$0{,}00E+00$	$-6{,}56E+00\dagger$
F19	$7{,}67E+00\dagger$	$-6{,}44E+00\dagger$
F20	$6{,}89E+00\dagger$	$-8{,}91E+00\dagger$
F21	$7{,}08E-01$	$-1{,}61E+04\dagger$
F22	$-2{,}46E+00\dagger$	$2{,}41E+00\dagger$
F23	$-3{,}74E+00\dagger$	$-1{,}90E+00$
F24	$-6{,}36E-01$	$-5{,}73E+00\dagger$
F25	$-9{,}08E+00\dagger$	$-1{,}10E+01\dagger$
F26	$-2{,}64E-01$	$-3{,}24E+00\dagger$
F27	$-5{,}54E+00\dagger$	$2{,}59E+00\dagger$
F28	$-1{,}23E+01\dagger$	$-9{,}74E+00\dagger$
F29	$-5{,}94E+00\dagger$	$-2{,}15E+00\dagger$
F30	$-1{,}93E+00$	$-6{,}41E+00\dagger$
F31	$6{,}62E+00\dagger$	$-9{,}66E+00\dagger$

TABLEAU 2.9: Les valeurs de la statistique t (en notation scientifique) d'un test t bilatéral avec 48 degrés de liberté et un seuil de signification de 0,05 († la différence entre les deux algorithmes est statistiquement significative)

nulle d'égalité des moyennes en faveur de l'hypothèse alternative et en conclure, avec une probabilité d'erreur de l'ordre de 5%, que les moyennes obtenues par *2-StageBBO-DE* et BBO sont significativement différentes. Pour les autres cas, aucune différence significative n'a été trouvée entre les deux algorithmes, au sens statistique du terme.

Nous pouvons conclure que les algorithmes comparés sont statistiquement différents dans la plupart des cas avec un degrés de confiance de 95%.

2.5.3.2 Étude de l'influence de l'opérateur de mutation

Nous désirons connaître l'apport des différentes stratégies de mutation, présentées dans la section 2.4.2.2, en terme de qualité de solutions trouvées. Pour cela nous allons comparer les

résultats obtenus à l'aide de différentes versions de l'algorithme *2-StageBBO-DE*, comportant des schémas de mutation différents. Le tableau 2.10 nous indique, pour chaque schéma, la valeur moyenne trouvée sur 25 exécutions pour chacune des versions de l'algorithme et l'écart type entre les différentes exécutions. Les valeurs en gras représentent les meilleurs résultats obtenus.

Pour évaluer les performances des nouvelles versions de l'algorithme *2-StageBBO-DE*, nous reprenons les expériences réalisées pour la version de l'algorithme utilisant le schéma de mutation *DE/Rand/1* et consignées dans le tableau 2.8.

La comparaison des différentes versions de l'algorithme nous a permis de désigner comme vainqueur l'algorithme avec la stratégie de mutation *DE/Rand/1* pour la qualité des solutions fournies. En effet, *2-StageBBO-DE*(Rand/1) obtient les meilleurs valeurs moyennes de la fonction objectif pour 21 problèmes sur les 31 étudiés. Ce schéma de mutation offre une plus grande amplitude du mouvement, favorisant ainsi l'exploration de l'espace de recherche. Ces résultats valident notre attente que le choix d'un opérateur de mutation aléatoire combiné à un taux de croisement proche de 1, favorisent la diversification. Dans la section qui suit, nous avons choisi d'employer cette variante de l'algorithme pour la comparaison avec d'autres algorithmes de la littérature. Afin de vérifier si les différences entre les différentes versions de l'algorithme *2-StageBBO-DE* sont statistiquement significatives, nous avons réalisé un test de Student bilatéral, avec un nombre de degrés de liberté égal à $\nu = 48$ et un seuil de signification $\alpha = 0{,}05$. Les résultats du test sont consignés dans le tableau 2.11. Ils sont résumés ci-après :

- ***2-StageBBO-DE(Rand/1)* vs. *2-StageBBO-DE(Best/1)* (1 vs. 2)** : La valeur calculée de la statistique t est plus grande que la valeur critique pour 22 fonctions de test. Nous pouvons donc rejeter l'hypothèse nulle, selon laquelle il n'y aurait pas de différence significative entre les deux algorithmes, avec une probabilité d'erreur de 5%.
- ***2-StageBBO-DE(Rand/1)* vs. *2-StageBBO-DE(Current to best)* (1 vs. 3)** : Dans près de la moitié des cas, la valeur de la statistique t calculée ne dépasse pas la valeur critique. Cela est particulièrement vrai pour les fonctions du *benchmark* CEC 2005 où, dans 12 cas sur les 17, l'hypothèse nulle ne peut être rejetée. Néanmoins, la différence entre les deux algorithmes est statistiquement significative pour les autres cas, particulièrement pour les fonctions standards.
- ***2-StageBBO-DE(Rand/1)* vs. *2-StageBBO-DE(Rand/2)* (1 vs. 4)** : La valeur calculée de la statistique t est inférieure à la valeur critique pour 5 fonction de test sur un total de 31 fonctions. Nous devons accepter l'hypothèse nulle $\mathcal{H}_0$ pour ces fonctions et nous ne pouvons donc prétendre qu'il existe une différence statistique significative entre les deux algorithmes. Pour les 26 fonctions restantes, la différence entre les deux algorithmes est significative d'un point de vue statistique.

Fonction	*2-StageBBO-DE(Best/1)*		*2-StageBBO-DE(Rand/1)*		*2-StageBBO-DE(Current to best)*		*2-StageBBO-DE(Rand/2)*		*2-StageBBO-DE(Best/2)*	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
F1	$2{,}24E-01$	$1{,}76E-01$	$\mathbf{7{,}76E-36}$	$\mathbf{3{,}80E-35}$	$8{,}87E-02$	$7{,}49E-02$	$6{,}90E-01$	$2{,}94E-01$	$3{,}20E-03$	$5{,}55E-03$
F2	$5{,}76E-02$	$2{,}92E-02$	$6{,}14E-04$	$2{,}64E-03$	$\mathbf{3{,}85E-04}$	$\mathbf{7{,}72E-04}$	$2{,}32E-01$	$4{,}47E-02$	$1{,}93E-02$	$7{,}54E-03$
F3	$2{,}55E+02$	$1{,}47E+02$	$\mathbf{1{,}52E-11}$	$\mathbf{1{,}86E-11}$	$8{,}63E+01$	$4{,}98E+01$	$7{,}22E+02$	$2{,}46E+02$	$1{,}63E+00$	$8{,}29E-01$
F4	$1{,}52E+00$	$2{,}78E-01$	$\mathbf{9{,}02E-02}$	$\mathbf{6{,}10E-02}$	$9{,}40E-01$	$1{,}69E-01$	$1{,}19E+00$	$1{,}46E-01$	$1{,}68E-01$	$6{,}31E-02$
F5	$4{,}88E+01$	$3{,}57E+01$	$6{,}71E+01$	$2{,}61E+01$	$5{,}95E+01$	$3{,}30E+01$	$5{,}88E+01$	$3{,}76E+01$	$\mathbf{4{,}75E+01}$	$\mathbf{3{,}47E+01}$
F6	$2{,}82E-01$	$3{,}03E-01$	$\mathbf{0{,}00E+00}$	$\mathbf{0{,}00E+00}$	$9{,}30E-02$	$7{,}93E-02$	$7{,}24E-01$	$3{,}24E-01$	$1{,}65E-03$	$1{,}18E-03$
F7	$2{,}79E-03$	$1{,}22E-03$	$1{,}54E-03$	$3{,}41E-04$	$\mathbf{1{,}13E-03}$	$\mathbf{4{,}93E-04}$	$1{,}51E-02$	$7{,}97E-03$	$2{,}73E-03$	$1{,}18E-03$
F8	$2{,}10E-01$	$2{,}91E-01$	$\mathbf{1{,}73E-01}$	$\mathbf{3{,}25E-01}$	$3{,}31E+05$	$1{,}28E+05$	$5{,}21E+13$	$2{,}47E+14$	$7{,}30E+05$	$5{,}35E+06$
F9	$8{,}61E-03$	$4{,}36E-03$	$\mathbf{2{,}24E-03}$	$\mathbf{3{,}25E-03}$	$1{,}22E-02$	$1{,}73E-02$	$2{,}43E-02$	$1{,}15E-02$	$2{,}03E-02$	$8{,}89E-03$
F10	$9{,}91E-02$	$3{,}97E-02$	$\mathbf{4{,}63E-03}$	$\mathbf{1{,}47E-02}$	$2{,}67E-02$	$2{,}65E-02$	$2{,}53E-01$	$8{,}40E-02$	$1{,}44E-02$	$8{,}16E-03$
F11	$2{,}29E-01$	$1{,}13E-01$	$\mathbf{1{,}33E-17}$	$\mathbf{6{,}53E-17}$	$9{,}44E-02$	$1{,}10E-01$	$3{,}81E-01$	$1{,}06E-01$	$2{,}47E-02$	$2{,}03E-02$
F12	$1{,}93E-03$	$3{,}19E-03$	$9{,}64E-06$	$3{,}68E-05$	$7{,}91E-04$	$3{,}06E-03$	$6{,}77E-03$	$9{,}13E-03$	$\mathbf{3{,}61E-06}$	$\mathbf{3{,}70E-06}$
F13	$1{,}07E-02$	$1{,}44E-02$	$\mathbf{4{,}51E-04}$	$\mathbf{2{,}15E-03}$	$6{,}20E-03$	$9{,}63E-03$	$3{,}77E-02$	$1{,}89E-02$	$1{,}47E-03$	$3{,}55E-03$
F14	$1{,}03E+06$	$1{,}38E+05$	$1{,}10E+06$	$1{,}14E+05$	$1{,}13E+06$	$1{,}16E+05$	$1{,}16E+06$	$1{,}37E+05$	$\mathbf{1{,}08E+06}$	$\mathbf{1{,}42E+05}$
F15	$\mathbf{0{,}00E+00}$	$\mathbf{0{,}00E+00}$	$\mathbf{0{,}00E+00}$	$\mathbf{0{,}00E+00}$	$\mathbf{0{,}00E+00}$	$\mathbf{0{,}00E+00}$	$8{,}32E-02$	$5{,}12E-02$	$4{,}60E-03$	$2{,}83E-03$
F16	$1{,}80E-01$	$4{,}52E-01$	$\mathbf{0{,}00E+00}$	$\mathbf{0{,}00E+00}$	$1{,}72E-01$	$4{,}35E-01$	$9{,}26E+01$	$5{,}38E+01$	$6{,}75E-01$	$6{,}89E-01$
F17	$1{,}44E+04$	$1{,}88E+04$	$\mathbf{1{,}14E-10}$	$\mathbf{1{,}28E-10}$	$2{,}52E+03$	$5{,}47E+03$	$9{,}75E+05$	$9{,}28E+05$	$2{,}73E+05$	$2{,}52E+05$
F18	$2{,}28E-11$	$1{,}08E-10$	$\mathbf{0{,}00E+00}$	$\mathbf{0{,}00E+00}$	$3{,}51E-06$	$1{,}72E-05$	$2{,}27E+02$	$2{,}03E+02$	$6{,}24E-01$	$1{,}05E+00$
F19	$1{,}36E+02$	$1{,}44E+02$	$\mathbf{1{,}41E-05}$	$\mathbf{9{,}16E-06}$	$8{,}33E+00$	$1{,}28E+01$	INF	INF	$2{,}04E+02$	$8{,}21E+01$
F20	$1{,}39E+01$	$2{,}13E+01$	$\mathbf{4{,}14E+00}$	$\mathbf{2{,}78E+00}$	$1{,}06E+01$	$1{,}40E+01$	$1{,}18E+02$	$6{,}73E+01$	$4{,}69E+02$	$2{,}01E+03$
F21	$1{,}26E+01$	$1{,}22E+01$	$\mathbf{6{,}42E-02}$	$\mathbf{8{,}79E-02}$	$1{,}96E+01$	$1{,}16E+01$	$1{,}22E+03$	$2{,}27E+02$	$9{,}80E-01$	$8{,}74E-02$
F22	$2{,}03E+01$	$6{,}97E-02$	$\mathbf{2{,}03E+01}$	$\mathbf{5{,}15E-02}$	$2{,}03E+01$	$7{,}99E-02$	$2{,}04E+01$	$7{,}99E-02$	$2{,}04E+01$	$8{,}27E-02$
F23	$1{,}48E-02$	$2{,}31E-02$	$\mathbf{4{,}24E-03}$	$\mathbf{1{,}54E-02}$	$8{,}10E-03$	$1{,}26E-02$	$4{,}60E-02$	$3{,}67E-02$	$3{,}69E-02$	$2{,}19E-02$
F24	$1{,}23E+01$	$6{,}62E+00$	$6{,}93E+00$	$3{,}28E+00$	$\mathbf{5{,}77E+00}$	$\mathbf{2{,}23E+00}$	$9{,}90E+00$	$4{,}26E+00$	$9{,}20E+00$	$3{,}99E+00$
F25	$2{,}75E+00$	$1{,}38E+00$	$\mathbf{1{,}08E+00}$	$\mathbf{1{,}36E+00}$	$1{,}09E+00$	$1{,}01E+00$	$5{,}05E+00$	$1{,}25E+00$	$3{,}70E+00$	$1{,}92E+00$
F26	$4{,}37E+02$	$6{,}13E+02$	$\mathbf{1{,}31E+02}$	$\mathbf{4{,}21E+02}$	$2{,}12E+02$	$4{,}92E+02$	$6{,}92E+02$	$8{,}89E+02$	$8{,}81E+02$	$1{,}33E+03$
F27	$4{,}70E-01$	$1{,}33E-01$	$3{,}62E-01$	$9{,}43E-02$	$3{,}77E-01$	$9{,}48E-02$	$\mathbf{3{,}41E-01}$	$\mathbf{1{,}04E-01}$	$4{,}04E-01$	$1{,}19E-01$
F28	$2{,}65E+00$	$5{,}78E-01$	$\mathbf{2{,}13E+00}$	$\mathbf{4{,}68E-01}$	$2{,}33E+00$	$4{,}56E-01$	$3{,}06E+00$	$3{,}22E-01$	$3{,}00E+00$	$2{,}93E-01$
F29	$2{,}10E+02$	$1{,}84E+02$	$6{,}40E+01$	$1{,}47E+02$	$1{,}78E+02$	$2{,}01E+02$	$\mathbf{6{,}36E+01}$	$\mathbf{1{,}46E+02}$	$2{,}14E+02$	$1{,}73E+02$
F30	$1{,}13E+02$	$1{,}44E+01$	$1{,}02E+02$	$7{,}31E+00$	$\mathbf{9{,}64E+01}$	$\mathbf{1{,}21E+01}$	$1{,}13E+02$	$8{,}97E+00$	$1{,}12E+02$	$1{,}15E+01$
F31	$1{,}05E+02$	$1{,}43E+01$	$9{,}90E+01$	$7{,}40E+00$	$1{,}03E+02$	$1{,}22E+01$	$1{,}08E+02$	$1{,}73E+01$	$\mathbf{9{,}67E+01}$	$\mathbf{1{,}69E+01}$

TABLEAU 2.10: Résultats des versions de *2-StageBBO-DE*, comportant des schémas de mutation différents, sur le *benchmark* des 31 fonctions, donnant pour chaque fonction la performance moyenne et l'écart type sur les 25 essais. Les résultats en gras indiquent le meilleur résultat ou l'optimum global.

– ***2-StageBBO-DE(Rand/1)* vs. *2-StageBBO-DE(Best/2)* (1 vs. 5)** : Dans la plupart des cas, 24 sur les 31 fonctions, la valeur t calculée dépasse la valeur critique, alors l'hypothèse nulle peut être rejetée avec un risque d'erreur de l'ordre de 5%. Le test t confirme, par conséquent, que les moyennes obtenues par les deux algorithmes sont significativement différentes, statistiquement parlant.

L'algorithme *2-StageBBO-DE* avec le schéma de mutation *DE/Rand/1* est statistiquement différent et plus performant que les autres versions de l'algorithme. Ces résultats confirment une fois de plus que le choix de l'opérateur de mutation a un effet significatif sur la performance de l'algorithme *2-StageBBO-DE*.

2.5.3.3 Comparaison de *2-StageBBO-DE* avec d'autres méthodes

Pour terminer l'étude de l'algorithme *2-StageBBO-DE*, nous allons évaluer sa pertinence face à différents algorithmes s'étant confrontés aux fonctions de *benchmark* ($F1$ à $F14$). Nous avons alors récupéré les résultats de 4 versions de l'algorithme BBO [Du et al., 2009]. Ces algorithmes sont utilisés dans la configuration de base proposée dans l'article de référence. Nous ne détaillerons pas ici chacune de ces méthodes, nous engageons donc le lecteur à se référer à l'article indiqué pour de plus amples informations. Nous avons, en plus, implémenté un algorithmes génétique (GA) pour les besoins de l'étude. Les différents algorithmes sont :

1. BBO original
2. BBO/ES : une hybridation de BBO avec la stratégie d'évolution (ES)
3. BBO/RE : BBO avec un mécanisme de refus d'immigration (*immigration refusal*)
4. BBO/ES/RE : une hybridation de BBO/ES avec BBO/RE
5. GA : un algorithme génétique qui implémente un croisement uniforme et une sélection par tirage à la roulette. Sur la base des expériences menées dans [DeJong, 1975], la taille de la population de GA est fixée à 100, la taille de la mémoire élite est réglée à 2 et les taux de croisement et de mutation sont fixés à 0,6 et 0,001 respectivement.

Les résultats numériques des comparaisons, en terme de meilleures valeurs de la fonction objectif, sont présentés dans le tableau 2.12. Les meilleures valeurs sont données en gras. La comparaison est établie à partir des performances des différents algorithmes sur les fonctions $F1$ à $F14$ en dimensions 20 [Simon, 2008]. 100 exécutions de chacun de ces algorithmes ont été effectuées, pour un nombre maximum de générations fixé à 100. Les résultats pour les fonctions $F15$ à $F31$ pour les quatre versions de BBO susmentionnées ne sont pas disponibles.

Il est observé, à partir des résultats obtenus, que l'algorithme *2-StageBBO-DE* se détache en obtenant les meilleurs résultats pour 11 fonctions sur un total de 14. Les résultats rapportés

Function	1 vs. 2	1 vs. 3	1 vs. 4	1 vs. 5
	t-test	t-test	t-test	t-test
F1	$-6{,}36E+00\dagger$	$-5{,}92E+00\dagger$	$-1{,}17E+01\dagger$	$-2{,}88E+00\dagger$
F2	$-9{,}74E+00\dagger$	$4{,}16E-01$	$-2{,}59E+01\dagger$	$-1{,}17E+01\dagger$
F3	$-8{,}69E+00\dagger$	$-8{,}67E+00\dagger$	$-1{,}47E+01\dagger$	$-9{,}84E+00\dagger$
F4	$-2{,}51E+01\dagger$	$-2{,}36E+01\dagger$	$-3{,}50E+01\dagger$	$-4{,}41E+00\dagger$
F5	$2{,}07E+00\dagger$	$9{,}10E-01$	$9{,}13E-01$	$2{,}26E+00\dagger$
F6	$-4{,}65E+00\dagger$	$-5{,}86E+00\dagger$	$-1{,}12E+01\dagger$	$-6{,}99E+00\dagger$
F7	$-4{,}92E+00\dagger$	$3{,}38E+00\dagger$	$-8{,}53E+00\dagger$	$-4{,}85E+00\dagger$
F8	$-4{,}28E-01$	$-1{,}29E+01\dagger$	$-1{,}05E+00$	$-6{,}82E-01$
F9	$-5{,}86E+00\dagger$	$-2{,}85E+00\dagger$	$-9{,}22E+00\dagger$	$-9{,}51E+00\dagger$
F10	$-1{,}11E+01\dagger$	$-3{,}63E+00\dagger$	$-1{,}46E+01\dagger$	$-2{,}90E+00\dagger$
F11	$-1{,}01E+01\dagger$	$-4{,}31E+00\dagger$	$-1{,}79E+01\dagger$	$-6{,}09E+00\dagger$
F12	$-3{,}01E+00\dagger$	$-1{,}28E+00$	$-3{,}70E+00\dagger$	$8{,}16E-01$
F13	$-3{,}53E+00\dagger$	$-2{,}91E+00\dagger$	$-9{,}81E+00\dagger$	$-1{,}23E+00$
F14	$1{,}95E+00$	$-7{,}43E-01$	$-1{,}58E+00$	$7{,}56E-01$
F15	$0{,}00E+00$	$0{,}00E+00$	$-8{,}13E+00\dagger$	$-8{,}13E+00\dagger$
F16	$-1{,}99E+00$	$-1{,}98E+00$	$-8{,}61E+00\dagger$	$-4{,}90E+00\dagger$
F17	$-3{,}81E+00\dagger$	$-2{,}31E+00\dagger$	$-5{,}25E+00\dagger$	$-5{,}42E+00\dagger$
F18	$-1{,}06E+00$	$-1{,}02E+00$	$-5{,}58E+00\dagger$	$-2{,}97E+00\dagger$
F19	$-4{,}74E+00\dagger$	$-3{,}26E+00\dagger$	$INF\dagger$	$-1{,}24E+01\dagger$
F20	$-2{,}26E+00\dagger$	$-2{,}27E+00\dagger$	$-8{,}46E+00\dagger$	$-1{,}16E+00$
F21	$-5{,}11E+00\dagger$	$-8{,}42E+00\dagger$	$-2{,}70E+01\dagger$	$-3{,}69E+01\dagger$
F22	$6{,}04E-02$	$1{,}85E+00$	$-5{,}57E+00\dagger$	$-3{,}13E+00\dagger$
F23	$-1{,}90E+00$	$-9{,}69E-01$	$-5{,}25E+00\dagger$	$-6{,}09E+00\dagger$
F24	$-3{,}62E+00\dagger$	$1{,}46E+00$	$-2{,}77E+00\dagger$	$-2{,}20E+00\dagger$
F25	$-4{,}30E+00\dagger$	$-2{,}30E-02$	$-1{,}07E+01\dagger$	$-5{,}56E+00\dagger$
F26	$-2{,}06E+00$	$-6{,}28E-01$	$-2{,}85E+00\dagger$	$-2{,}68E+00\dagger$
F27	$-3{,}29E+00\dagger$	$-5{,}58E-01$	$7{,}46E-01$	$-1{,}39E+00$
F28	$-3{,}54E+00\dagger$	$-1{,}53E+00$	$-8{,}20E+00\dagger$	$-7{,}95E+00\dagger$
F29	$-3{,}10E+00\dagger$	$-2{,}29E+00\dagger$	$1{,}13E-02$	$-3{,}29E+00\dagger$
F30	$-3{,}33E+00\dagger$	$2{,}04E+00$	$-4{,}52E+00\dagger$	$-3{,}75E+00\dagger$
F31	$-1{,}84E+00$	$-1{,}32E+00$	$-2{,}39E+00\dagger$	$6.21E-01$

TABLEAU 2.11: Les valeurs de la statistique t (en notation scientifique) d'un test t bilatéral avec 48 degrés de liberté et un seuil de signification de 0,05, entre *2-StageBBO-DE* avec la stratégie de mutation DE/Rand/1 et les quatre autres variantes. Les notations 1, 2, 3, 4 et 5 sont utilisées pour désigner respectivement les versions *2-StageBBO-DE*(Rand/1), *2-StageBBO-DE*(Best/1), *2-StageBBO-DE*(Current to best), *2-StageBBO-DE*(Rand/2) et *2-StageBBO-DE*(Best/2) († la différence entre les deux algorithmes est statistiquement significative)

Fonction	BBO	BBO/ES	BBO/RE	BBO/RS/RE	GA	*2-StageBBO-DE*
F1	0,16	**0,00**	0,12	0,01	0,978	**0,00**
F2	0,80	0,10	0,70	0,10	6,700	**0,05**
F3	51,41	9,52	28,69	12,10	3759,920	**4,90**
F4	680,93	654,65	866,16	889,69	628,000	**333,39**
F5	17,83	12,8	21,41	**13,44**	103,549	15,34
F6	62,00	7,00	39,00	7,00	822,000	**1,62**
F7	$3{,}68E-04$	$4{,}81E-06$	$2{,}22E-04$	$6{,}33E-06$	$3{,}93E-03$	$\mathbf{1{,}32E-07}$
F8	10,70	8,40	10,50	9,30	415,414	**7,85**
F9	1,93	**0,00**	4,04	**0,00**	28,151	2,90
F10	3,56	1,34	3,03	1,42	12,746	**0,20**
F11	1,40	1,04	1,42	1,07	3,993	**0,63**
F12	1,05	0,04	1,10	0,03	3,430	**0,01**
F13	4,07	0,46	4,56	0,51	26,503	**0,21**
F14	9570,1	4503,96	6216,63	**2248,52**	25807,445	2529,70

TABLEAU 2.12: Résultats sur le *benchmark* des 14 fonctions, donnant pour chaque fonction la meilleure performance sur les 100 essais. Les résultats en gras indiquent le meilleur résultat ou l'optimum global ($D = 20$)

dans [Du et al., 2009], sont meilleurs pour les fonctions *Rosenbrock*, *Rastrigin* et *Fletcher-Powell*. En revanche, l'algorithme GA offre de moins bons résultats sur toutes les fonctions de test. Ces résultats viennent, encore une fois, confirmer la pertinence de la méthode proposée.

2.6 Conclusion

Dans ce chapitre, nous avons rappelé le principe de fonctionnement de l'algorithme BBO et testé l'influence de ses différents paramètres sur la qualité des résultats obtenus. Nous avons pu constaté, à travers cette analyse, que BBO souffre d'un manque de diversité, qui conduit souvent à des solutions sous-optimales. Nous avons par la suite présenté un nouvel algorithme, appelé *2-StageBBO-DE*, qui est une hybridation de l'algorithme BBO, auquel nous avons ajouté un opérateur de sélection, et l'algorithme à évolution différentielle (DE). Les performances de l'algorithme proposé sont testées sur plusieurs fonctions de *benchmark* et confrontées aux algorithmes BBO et DE classiques. Les expériences réalisées montrent que *2-StageBBO-DE* s'impose comme la meilleure solution dans la plupart des cas.

Chapitre Trois

Élaboration des CBBOs : Algorithmes d'optimisation sous contraintes

3.1 Introduction

Nous avons présenté dans le chapitre 1 (section 1.8) les principales stratégies de prise en compte des contraintes dans les problèmes d'optimisation. Comme nous l'avons mentionné, pour résoudre un problème d'optimisation, deux types d'approches peuvent être utilisés à savoir, les approches exactes et les approches stochastiques. Les approches exactes garantissent l'optimalité des solutions trouvées. Cependant, leur utilisation est souvent limitée à des problèmes de petite taille, car, trouver les solutions optimales a un coût exponentiel dans le pire des cas. Les méthodes stochastiques sont une alternative aux algorithmes exacts. Ces méthodes, notamment les algorithmes évolutionnaires, ont fait l'objet de nombreux travaux pour prendre en compte les contraintes. Parmi les approches développées, les plus utilisées sont celles basées sur les fonctions de pénalité, vue la facilité de leur implémentation et parce qu'elles donnent souvent de bons résultats. Nous nous inscrivons dans cette catégorie de méthode pour la prise en compte des contraintes dans l'algorithme BBO.

L'algorithme BBO a été à l'origine surtout utilisé pour les problèmes d'optimisation sans contraintes, où des modifications ont été apportées afin d'en améliorer les performances. Nous pouvons citer à titre d'exemple les travaux réalisés par *Du et al.* [Du et al., 2009], *Ergezer et al.* [Ergezer et al., 2009], *Gong et al.* [Gong et al., 2010a,b] et *Boussaïd et al.* [Boussaïd et al., 2011a]. Par contre, très peu de travaux ont été menés dans le cadre de l'optimisation avec contraintes. Citons à titre d'exemple [Ma & Simon, 2011].

Dans ce travail, notre objectif principal est d'étendre le domaine d'application de l'algorithme BBO afin qu'il puisse également être utilisé comme une stratégie d'optimisation stochastique pour résoudre des problèmes d'optimisation sous contraintes, où il est question de trouver une solution pour une fonction objectif donnée, en tenant compte des contraintes d'inégalité et/ou d'égalité.

Ce chapitre présente trois variantes de l'algorithme BBO [Boussaïd et al., 2012]. Dans la première variante, nommée CBBO (pour *Constrained BBO*), trois composantes principales viennent s'ajouter à l'algorithme BBO de base pour la prise en compte des contraintes :

1. Une *fonction de pénalisation*, utilisée afin de défavoriser les solutions non admissibles. Il est en effet plus intéressant de conserver, tout en pénalisant, les individus non admissibles car ils peuvent permettre de générer des individus admissibles de bonne qualité. La fonction objectif prend donc en compte les violations de contraintes en pénalisant les individus concernés.
2. Une *stratégie de classement stochastique* qui permet de classer les individus de la population afin de prendre en compte les contraintes de manière stochastique.
3. Une *stratégie de sélection* basée sur le principe de la supériorité des solutions faisables. Ainsi, toute solution faisable est meilleure que toute solution infaisable.

Un autre algorithme, résultant de l'hybridation de CBBO avec DE, nommé CBBO-DE est aussi proposée dans le cadre de cette étude. Ce dernier incorpore l'opérateur de mutation emprunté aux algorithmes d'évolution différentielle [Storn & Price, 1997]. Enfin, nous proposons un nouvel opérateur nommé « *Pivot-2* » qui sera implémenté dans la variante CBBO-PM (*CBBO with Pivot Method*). Les trois algorithmes susmentionnés utilisent les mêmes mécanismes de prise en compte des contraintes, ils diffèrent seulement par leur façon de faire évoluer la population à travers les opérateurs de variation qui seront présentés plus en détail dans la section 3.2.3.

Dans ce chapitre, nous allons décrire en détail les trois algorithmes proposés, en présenter une analyse expérimentale, ainsi qu'une comparaison avec d'autres méthodes d'optimisation de la littérature afin d'estimer leurs performances.

3.2 *Constrained BBOs*

Les idées principales de notre approche sont les suivantes. Nous considérons à la base l'algorithme BBO auquel nous allons associer une stratégie de prise en compte des contraintes, basée sur le principe de pénalisation, empêchant les individus de devenir incohérents ; une stratégie de classement stochastique des individus et une stratégie de sélection basée sur la supériorité des individus faisables. Les étapes de l'algorithme CBBO sont illustrées dans la figure 3.1.

Une première variante de l'algorithme CBBO est appelée CBBO-DE. Cette dernière, associe à l'opérateur de migration de BBO et à un opérateur de sélection spécifique capable de prendre en compte les contraintes, l'opérateur de mutation issu des algorithmes d'évolution différentielle (DE).

Une autre variante, CBBO-PM, intègre un nouvel opérateur appelé « *Pivot-2* » dans le processus d'évolution de la population. Nous détaillerons dans ce qui suit le principe de fonctionnement des trois méthodes issues des réflexions présentées dans l'introduction. Les trois

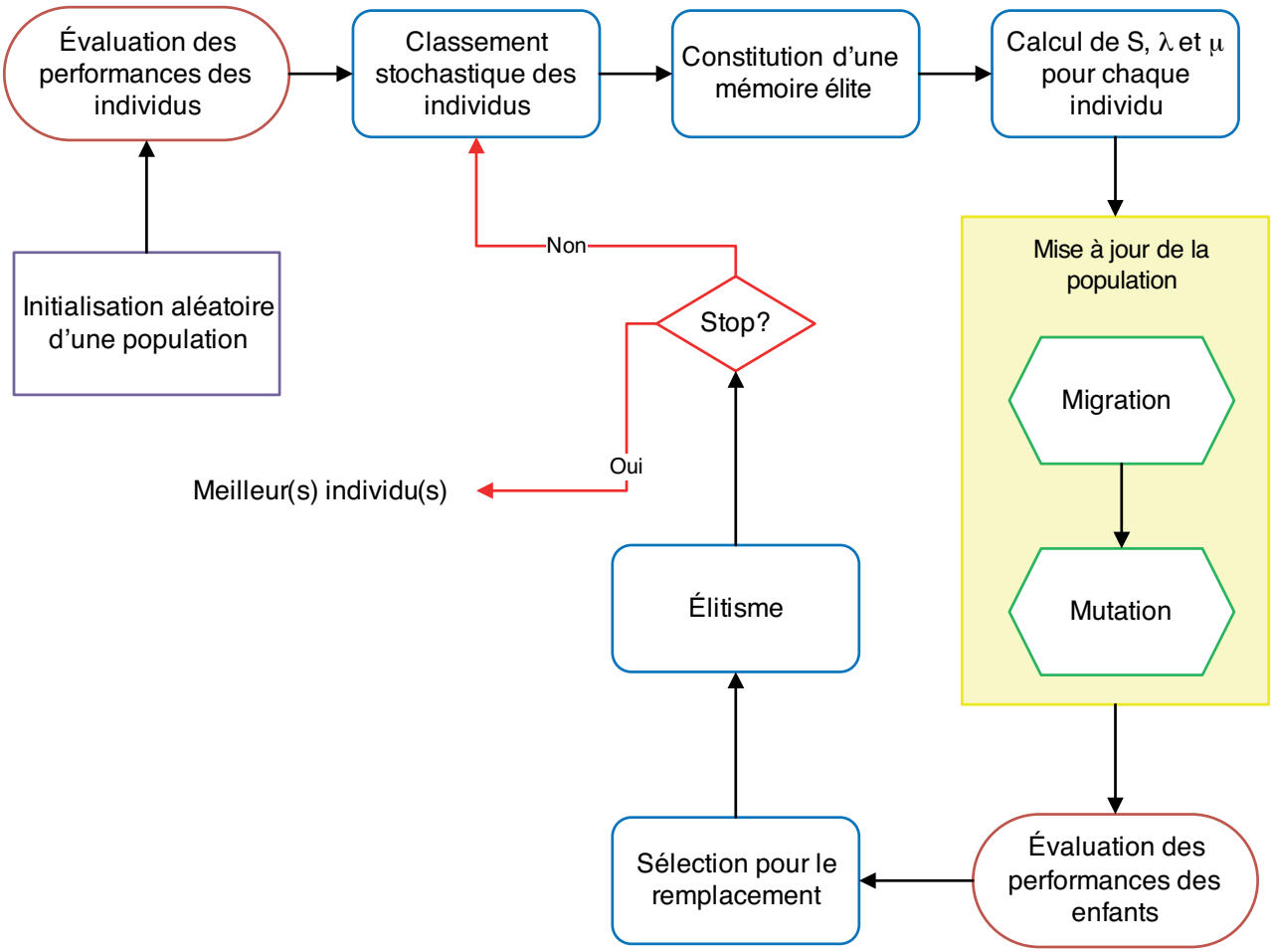


FIGURE 3.1: Principe de l'algorithme CBBO

algorithmes utilisent les mêmes mécanismes de prise en compte des contraintes, ils diffèrent seulement par leur façon de faire évoluer la population à travers les opérateurs de variation qui seront présentés plus en détail dans la section 3.2.3.

3.2.1 Prise en compte des contraintes

Commençons par rappeler la définition d'un problème d'optimisation avec contraintes :

$$\begin{cases} \text{Trouver } x \text{ qui minimise} & f(x) \\ \text{sous les contraintes} & g_i(x) \leq 0, \quad i = 1, 2, \ldots, p \\ & h_j(x) = 0, \quad j = 1, 2, \ldots, q \end{cases} \tag{3.2.1}$$

où f, g_i et h_j sont des fonctions réelles dans l'espace des solution $\mathcal{S}$. Rappelons aussi que l'espace des solutions réalisables du problème (3.2.1), noté $\mathcal{F} \subseteq \mathcal{S}$, représente l'ensemble des solutions qui satisfont toutes les contraintes du problème considéré. L'ensemble $\mathcal{F}$ est défini par :

$$\mathcal{F} = \{x \in \mathcal{S} : \ g_i(x) \geq 0, \ i = 1, \ldots, p; \ h_j(x) = 0, j = 1, \ldots, q\} \tag{3.2.2}$$

Nous considérons seulement les contraintes d'inégalité de type « supérieur ou égal ». Les contraintes de type « inférieur ou égal » peuvent être transformées en ces premières en appliquant le même principe de dualité, qui permet de transformer les objectifs de maximisation en ceux de minimisation.

Le principe adopté dans ce travail consiste à se ramener à un problème sans contraintes en associant à la fonction objectif une fonction de pénalité. Alors le problème sans contraintes à optimiser peut être formulé par :

$$Fitness(x) = f(x) + Penalty(x) \tag{3.2.3}$$

La fonction de pénalité, notée $Penalty$, utilise la mesure des violations des contraintes, et elle est calculée par :

$$F_{penalty}(x) = \sum_{j=1}^{m} \delta_j(x) \tag{3.2.4}$$

où $m = p + q$ est le nombre total de contraintes et $\delta_j(x)$ $(j = 1, \ldots, m)$ représentent la quantité et/ou le volume des contraintes violées. Par définition, la pénalité est nulle si x est admissible. La mesure de violation des contraintes est calculée comme suit :

$$\delta_j(x) = \begin{cases} \max[0, g_j(x)]^{\alpha} & j = 1, \ldots p \\ |h_j(x)|^{\beta} & j = p+1, \ldots m \end{cases} \tag{3.2.5}$$

où α et β sont généralement fixés à 1 ou 2.

L'utilisation de la mesure de violation des contraintes dans la pénalisation aide à différencier les individus infaisables. Ainsi, les individus qui ont un taux de violation très élevés sont pénalisés plus que ceux qui ont de faibles violations.

3.2.2 Le classement stochastique des individus

Un des problèmes les plus importants dans l'optimisation sous contraintes est de savoir comment concilier la recherche entre les régions faisables et infaisables. Dans ce travail, nous mettons en œuvre la procédure de classement stochastique (*stochastic ranking*) [Runarsson & Yao, 2000] pour équilibrer dynamiquement la domination des individus faisables et infaisables. Chaque individu de la population est classé en fonction à la fois de la valeur de la fonction objectif et du degré de violation des contraintes. La procédure de migration se base sur ce classement pour partager les informations entre les différents individus.

La procédure de classement stochastique est donnée par l'algorithme 3.1, où N est le nombre de balayages en passant par toute la population, NP représente la taille de la population, $rand(0,1)$ est un générateur de nombres aléatoires uniformes. Dans cet algorithme, f est la fonction objectif qui doit être minimisée et $Penalty$ est le terme de pénalisation. Le paramètre

P_f est responsable de l'effet aléatoire du classement. Il sert à décider d'utiliser la fonction objectif ou la fonction de pénalisation pendant la comparaison. Ainsi deux individus adjacents $\vec{X}_j$ et $\vec{X}_{j+1}$, où au moins un est infaisable, ont une probabilité P_f d'être comparés selon leurs valeurs de la fonction objectif, et une probabilité $(1 - P_f)$ d'être comparés selon leurs taux de violation des contraintes. Dans le cas où les deux individus sont faisables, $P_f = 1$. Le degré de violation de contraintes d'une solution donnée $\vec{X}_j$ est donnée par l'équation (3.2.4).

Algorithme 3.1: Classement stochastique (*Stochastic ranking algorithm*)

```
1  X_j = j ∀j ∈ {1,...,NP}
2  pour i = 1 à N faire
3  |  pour j = 1 à NP - 1 faire
4  |  |  Tirer aléatoirement un nombre u ∈ rand(0,1)
5  |  |  si (Penalty(X_j) = Penalty(X_{j+1}) = 0) ou (u < P_f) alors
6  |  |  |  si (f(X_j) > f(X_{j+1})) alors
7  |  |  |  |  Permuter (X_j, X_{j+1})
8  |  |  |  sinon
9  |  |  |  |  si (Penalty(X_j) > Penalty(X_{j+1})) alors
10 |  |  |  |  |  Permuter (X_j, X_{j+1})
11 |  |  |  |  fin
12 |  |  |  fin
13 |  |  fin
14 |  fin
15 |  si Aucune permutation faite alors
16 |  |  break
17 |  fin
18 fin
```

3.2.3 Opérateurs de variation

Comme il a déjà été mentionné dans l'introduction, les algorithmes proposés adoptent des mécanismes différents pour mettre à jour la population. Les opérateurs de variation employés par chacun des algorithmes CBBO, CBBO-DE et CBBO-PM sont décrits ci-après :

1. **CBBO** : Dans l'algorithme CBBO, la population est mise à jour en appliquant successivement la procédure de migration suivie de la procédure de mutation, de façon similaire à l'algorithme BBO de base (cf. section 1.6.3.4 – algorithme 1.1).

2. **CBBO-DE** : L'algorithme CBBO-DE intègre la procédure de mutation héritée de l'algorithme DE [Storn & Price, 1997; Price et al., 2005], pour remplacer la procédure de

mutation de BBO. Contrairement à CBBO, CBBO-DE génère tout d'abord les vecteurs mutants, auxquels il applique l'opérateur de migration (cf. Algorithme 2.1).

(a) **DE Mutation**

Un mutant est créé en ajoutant la différence pondérée de deux autres individus de la population à un tiers vecteur comme le montre l'équation (3.2.6).

$$\vec{V}_{i,g} = \vec{X}_{r1,g} + F(\vec{X}_{r2,g} - \vec{X}_{r3,g}); \qquad r1 \neq r2 \neq r3 \tag{3.2.6}$$

Le cœfficient F, est le cœfficient de mutation qui permet de contrôler l'amplitude des mutations. Différentes stratégies de mutation existent, elles sont présentées plus en détail dans la section 2.4.2.2. Dans l'équation (3.2.6), il s'agit de la stratégie *DE/rand/1/bin*.

(b) **Migration**

Un processus de migration permet ensuite de créer une nouvelle solution $\vec{M}_{i,g}$ à partir du vecteur mutant $\vec{V}_{i,g}$ nouvellement créé et d'autres vecteurs de la population choisis aléatoirement. Cet opérateur permet d'assigner au $j^{\text{ème}}$ SIV de la nouvelle solution $\vec{M}_{i,g}$, suivant son taux d'immigration λ_i, soit le $j^{\text{ème}}$ SIV du vecteur mutant $\vec{V}_{i,g}$ ou bien celui d'un vecteur $\vec{V}_{k,g}$, choisi aléatoirement avec une probabilité proportionnelle à son taux d'émigration μ_k (équation (3.2.7)).

$$M_{i,j,g} = \begin{cases} V_{k,j,g} & \text{si } rand(0,1) < \lambda_i \\ V_{i,j,g} & \text{sinon} \end{cases} \tag{3.2.7}$$

où $i = 1, 2, \ldots, NP$, $j = 1, .., D$ avec D la dimension du problème d'optimisation. $V_{k,j,g}$ est la $j^{\text{ème}}$ variable de décision de l'individu $\vec{V}_{k,g}$. $v_{k,g}$ est choisi avec une probabilité en fonction de son taux d'émigration μ_k et le taux d'immigration λ_i du vecteur $M_{i,g}$.

3. **CBBO-PM**

L'algorithme CBBO-PM procède de manière similaire à CBBO-DE, mais à la place de la mutation applique un nouvel opérateur, nommé *Pivot-2*.

(a) **Méthode Pivot-2**

Cette méthode est inspirée des travaux présentés dans [Clerc, 2003]. Cependant, *Pivot-2* se déroule de manière différente. La stratégie principale de la méthode proposée est de générer un nouveau vecteur, pour un individu donné $\vec{X}_{i,g}$, à partir du meilleur individu de la population et d'un individu aléatoirement choisi dans le reste

de la population. Deux hypersphères sont alors définies. Leurs centres correspondent aux deux individus sélectionnés et ont le même rayon qui est égal à leur distance. La nouvelle solution, notée $\vec{V}_{i,g}$, est calculée comme la somme pondérée de deux vecteurs choisis aléatoirement dans les deux hypersphères, comme suit :

$$\vec{V}_{i,g} = c_1.alea_{Sphere}(H_{\vec{X}_{best,g}}) + c_2.alea_{Sphere}(H_{\vec{X}_{r,g}}) \tag{3.2.8}$$

où l'indice r est choisi aléatoirement dans l'intervalle $[1, NP]$ et doit être différent de l'indice courant i ; $\vec{X}_{best,g}$ est le meilleur individu de la population à la génération g, $alea_{Sphere}(H_{\vec{X}_{best,g}})$ est un nombre choisi aléatoirement, de manière uniforme, dans l'hypersphère de centre $\vec{X}_{best,g}$ et de rayon $\left\|\vec{X}_{best,g} - \vec{X}_{r,g}\right\|$ et $alea_{Sphere}(H_{\vec{X}_{r,g}})$ est un nombre aléatoire uniformément distribué dans l'hypersphère de centre $\vec{X}_{r,g}$ et de rayon $\left\|\vec{X}_{best,g} - \vec{X}_{r,g}\right\|$; $c_1 = \frac{f(\vec{X}_{best,g})}{f(\vec{X}_{best,g})+f(\vec{X}_{r,g})}$ et $c_2 = \frac{f(\vec{X}_{r,g})}{f(\vec{X}_{best,g})+f(\vec{X}_{r,g})}$.

La méthode *Pivot-2* est illustrée dans l'algorithme 3.2.

(b) **Migration**

La méthode Pivot, crée de nouveaux vecteurs $\vec{V}_{i,g}$ $(i = 1, \ldots, NP)$, qui sont à leur tour modifiés par l'opérateur de migration, comme décrit dans l'équation (3.2.7).

Algorithme 3.2: Pivot-2

pour $i = 1$ à NP **faire**
  Sélectionner aléatoirement $r \in [1, NP]$; $r \neq i$
  $distance = \left|(f(\vec{X}_{best,g}) + f(\vec{X}_{r,g})\right|$
  calculer c_1 et c_2 : $c_1 = f(\vec{X}_{best,g})/distance$ et $c_2 = 1 - c_1$
  pour $j = 1$ à D **faire**
    $radius = \left|\vec{X}_{best,j,g} - \vec{X}_{r,j,g}\right|$
    $POS_{1j} = rand(\vec{X}_{best,j,g} - radius, \vec{X}_{best,j,g} + radius)$
    $POS_{2j} = rand(\vec{X}_{r,j,g} - radius, \vec{X}_{r,j,g} + radius)$
    $V_{i,j,g} = c_1.POS_{1j} + c_2 POS_{2j}$
  fin
fin

3.2.4 Sélection

L'opérateur de sélection se base sur le principe de la supériorité des solutions faisables, afin de préserver les meilleurs individus pour les générations suivantes. Un individu de la population qui viole une contrainte se verra attribuer une mauvaise *fitness* et aura une probabilité forte d'être éliminé par le processus de sélection. Nous adoptons la méthode de sélection par tour-

noi binaire proposée par *Deb* [Deb, 2000], où deux individus sont comparés selon les critères suivants :

1. toute solution faisable est meilleure que toute solution infaisable
2. parmi deux solutions faisables, celle ayant la meilleure performance est sélectionnée
3. parmi deux solutions infaisables, celle ayant le plus petit degré de violation est sélectionnée.

Par conséquent, une solution $\vec{X}_{i,g}$ sera remplacée par sa version modifiée par les opérateurs de variation $\vec{M}_{i,g}$, pour survivre pour la prochaine génération $(g+1)$, selon le scénario suivant :

1. $\vec{X}_{i,g}$ est infaisable, mais $\vec{M}_{i,g}$ est faisable,
2. $\vec{X}_{i,g}$ et $\vec{M}_{i,g}$ sont faisables, mais $f(\vec{M}_{i,g}) < f(\vec{X}_{i,g})$,
3. $\vec{X}_{i,g}$ et $\vec{M}_{i,g}$ sont infaisables, mais $P(\vec{M}_{i,g}) < P(\vec{X}_{i,g})$.

Afin d'attribuer le même degré d'importance à toutes les contraintes, la valeur maximale du degré de violation de chaque contrainte, pour l'ensemble de la population, est utilisée pour normaliser chaque contrainte violée. La nouvelle mesure de pénalité est formulée comme suit :

$$P(\vec{X}_{i,g}) = \sum_{j=1}^{m} \frac{\delta_j(\vec{X}_{i,g})}{\delta_{\max}(j)} \tag{3.2.9}$$

où $\delta_j(\vec{X}_{i,g})$ est le degré de violation de la contrainte j par le $i^{\text{ème}}$ individu $\vec{X}_{i,g}$ (cf. équation 3.2.5) et $\delta_{\max}(j) = \max[\delta_j(\vec{X}_{i,g})]$ $(i = 1, \ldots, NP)$ est la valeur maximale du degré de violation de la $j^{\text{ème}}$ contrainte. Ainsi, la performance d'un individu infaisable ne dépend pas seulement de ses propres degrés de violation des contraintes, mais aussi de la population courante. La procédure de sélection est présentée dans l'algorithme 3.3.

Algorithme 3.3: Sélection

```
Comparer X⃗_{i,g} avec le vecteur mutant M⃗_{i,g}
si X⃗_{i,g} et M⃗_{i,g} sont faisables (i.e., P(X⃗_{i,g}) = P(M⃗_{i,g}) = 0) alors
    si (f(M⃗_{i,g}) < f(X⃗_{i,g})) alors
        X⃗_{i,g+1} = M⃗_{i,g}
    sinon
        X⃗_{i,g+1} = X⃗_{i,g}
    fin
sinon
    si X⃗_{i,g} est faisable et M⃗_{i,g} est infaisable alors
        X⃗_{i,g+1} = X⃗_{i,g}
    sinon
        si X⃗_{i,g} et M⃗_{i,g} sont faisables alors
            si (P(M⃗_{i,g}) < P(X⃗_{i,g})) alors
                X⃗_{i,g+1} = M⃗_{i,g}
            sinon
                X⃗_{i,g+1} = X⃗_{i,g}
            fin
        fin
    fin
fin
```

3.2.5 Elitisme

Une mémoire élite est ajoutée aux algorithmes proposés, pour préserver les meilleures solutions. Nous souhaitons, en effet, conserver des ensembles d'attributs présentant une performance que nous pouvons considérer comme bonne mais qui seraient perdus au cours des générations. La mémoire élite de taille n_{elit} est ordonnée en appliquant la procédure de classement stochastique (cf. algorithme 3.1).

3.3 Résultats numériques

Nous désirons connaître l'apport des différents mécanismes, présenté dans la section précédente, en terme de qualité de solutions trouvées. Pour cela nous allons comparer les résultats obtenus à l'aide de différentes versions de l'algorithme BBO, sur un ensemble de problèmes avec contraintes issu du *benchmark* proposé dans [Runarsson & Yao, 2000]. Ensuite, des résultats issus de la littérature sur les fonctions du même *benchmark* seront présentés et comparés avec notre approche.

Fonction	n	Type	ρ	IL	IN	EL	EN	a
g_{01}	13	Quadratic	0,0111%	9	0	0	0	6
g_{02}	20	Nonlinear	99,9971%	0	2	0	0	1
g_{03}	10	Nonlinear	0,0000%	0	0	0	1	1
g_{04}	5	Quadratic	52,1230%	0	6	0	0	2
g_{05}	4	Cubic	0,0000%	2	0	0	3	3
g_{06}	2	Nonlinear	0,0066%	0	2	0	0	2
g_{07}	10	Quadratic	0,0003%	3	5	0	0	6
g_{08}	2	Nonlinear	0,8560%	0	2	0	0	0
g_{09}	7	Nonlinear	0,5121%	0	4	0	0	2
g_{10}	8	Linear	0,0010%	3	3	0	0	6
g_{11}	2	Quadratic	0,0000%	0	0	0	1	1
g_{12}	3	Quadratic	4,7713%	0	1	0	0	0

TABLEAU 3.1: Résumé des 12 cas de test. Le ratio $\rho = |\mathcal{F}| / |\mathcal{S}|$ est le rapport entre la taille de l'espace faisable $\mathcal{F}$ et celle de l'espace de recherche $\mathcal{S}$. *IL* et *IN* représentent le nombre des inégalités linéaires et des inégalités non linéaires, respectivement ; *EL* et *EN* représentent le nombre des égalités linéaires et des égalités non linéaires, respectivement et a représente le nombre de contraintes actives à l'optimum

3.3.1 Fonctions de test

Les expériences ont été effectuées sur 12 fonctions sélectionnées dans l'ensemble des cas de test proposé dans [Runarsson & Yao, 2000]. La définition complète de ces fonctions est donnée en annexe. Chacune des méthodes proposées a été testée sur les fonctions de référence présentées dans le tableau 3.1. Les fonctions $g02$, $g03$, $g08$, et $g12$ sont des problèmes de maximisation que nous transformons en problèmes de minimisation en multipliant la fonction objectif par -1.

3.3.2 Paramétrage

Nous présentons dans cette section les paramètres des différentes versions de l'algorithme BBO. Le nombre d'individus dans la population est fixé à 100. Le nombre maximal de génération est de 3 500 générations (soit 350 000 évaluations de la fonction objectif) [Runarsson & Yao, 2000]. Pour CBBO, la mutation est appliquée avec une probabilité de 0,01. L'algorithme CBBO-DE, utilise la stratégie de mutation *DE/rand/1/bin* avec $F = 0{,}5$ [Storn & Price, 1997]. Ces paramètres sont illustrés dans le tableau 3.2. Pour l'algorithme de classement stochastique, la probabilité d'utiliser uniquement la fonction objectif pour la comparaison est choisie comme $P_f = 0{,}45$. Les paramètres α et β pour mesurer les taux de violation des contraintes sont fixés à 2.

Paramètres	**Notation**	**Value**
Taille de la population	NP	100
Paramètre d'élitisme	n_{elit}	2
Taux maximum d'immigration	I	1
Taux maximum d'émigration	E	1
Probabilité de mutation	m	0,01
Constante différentielle	F	0,5
Stratégie de mutation	*DE/Rand/1/bin*	

TABLEAU 3.2: Paramètres des différents algorithmes testés

3.3.3 Résultats expérimentaux et discussions

Pour effectuer les tests de comparaison sur les différents algorithmes définis précédemment, 30 exécutions de chacun de ces algorithmes ont été effectuées, pour un nombre maximum de générations fixé à 3 500. Nous avons réalisé, pour chaque cas de test, des statistiques descriptives des résultats trouvés. Ainsi, nous indiquons pour chaque fonction de test la moyenne sur les exécutions du meilleur individu (*Mean*), le meilleur sur les 30 exécutions (*Best*), la pire (*Worst*) solution sur les 30 exécutions, et l'écart type (*Std.*) entre les différentes exécutions. Les résultats des différentes expériences sont résumés dans le tableau 3.3. Toutes les solutions trouvées pour chaque cas sont faisables, exception faite pour le problème $g06$, où la solution se trouve sur la frontière du domaine faisable et dont aucune solution faisable n'a été trouvée par CBBO et CBBO-PM.

Pour l'algorithme CBBO, l'optimum exact a été trouvé pour 2 problèmes ($g08$ et $g12$) parmi les 12 étudiés. Pour les autres problèmes, les solutions obtenues sont proches de l'optimum global sauf pour $g06$ où aucune solution faisable n'a été trouvée. L'algorithme CBBO-PM a trouvé l'optimum exact pour 4 sur les 12 cas étudiés.

L'algorithme CBBO-DE a été en mesure de trouver l'optimum global pour 9 problèmes ($g01$, $g03$, $g04$, $g05$, $g06$, $g08$, $g09$, $g11$ and $g12$) et il a trouvé des solutions très proches de l'optimum global pour les trois autres problèmes ($g02$, $g07$, $g10$). Nous pouvons également noter que les écarts-types sont relativement faibles pour l'ensemble des essais, ce qui signifie que les valeurs obtenues sont regroupées autour de la moyenne. Pour plus de clarté, les solutions obtenues, qui sont de meilleure qualité, ont été mises en gras.

La conclusion que nous pouvons faire à partir du tableau 3.3 est que CBBO-DE a donné des résultats très satisfaisants pour toutes les fonctions testées et surpasse les deux autres algorithmes concurrents sur toutes les fonctions de test.

Fonction & Optimum	Statistiques	Algorithmeçà		
		CBBO	CBBO-PM	CBBO-DE
g01	Best	−14,989	**−15,000**	**−15,000**
	Mean	−14,957	**−15,000**	**−15,000**
−15,000	Worst	−14,919	**−15,000**	**−15,000**
	Std,	$1{,}80E-02$	$4{,}38E-14$	$8{,}21E-14$
g02	Best	−0,792776	−0,803035	**−0,803557**
	Mean	−0,777357	−0,758341	**−0,802774**
−0,803619	Worst	−0,754746	−0,670224	**−0,792576**
	Std,	$9{,}36E-03$	$3{,}64E-02$	$2{,}72E-03$
g03	Best	−0,999	−0,999470	**−1,000**
	Mean	−0,993	−0,995	**−1,000**
−1,000	Worst	−0,969	−0,983	**−1,000**
	Std,	$6{,}80E-03$	$4{,}13E-03$	$6{,}04E-16$
g04	Best	−30609,681	−30653,259	**−30665,539**
	Mean	−30361,016	−30513,266	**−30665,539**
−30665,539	Worst	−30005,392	−30319,140	**−30665,539**
	Std,	$1{,}38E+02$	$8{,}81E+01$	$1{,}67E-11$
g05	Best	5128,418	5125232	**5126,498**
	Mean	5201,248	5240,255	**5126,498**
5126,498	Worst	5361,189	5625,784	**5126,498**
	Std,	$6{,}69E+01$	$1{,}26E+02$	$2{,}23E-04$
g06	Best	NF	NF	**−6961,814**
	Mean	NF	NF	**−6961,814**
6961,814	Worst	NF	NF	**−6961,814**
	Std,	NF	NF	$4{,}55E-12$
g07	Best	25,206	27,890	**24,326**
	Mean	29,910	58,092	**24,345**
24,306	Worst	41,854	199,667	**24,378**
	Std,	$4{,}25E+00$	$4{,}05E+01$	$1{,}33E-02$
g08	Best	**−0,095825**	**−0,095825**	**−0,095825**
	Mean	**−0,095825**	**−0,095825**	**−0,095825**
−0,095825	Worst	−0,095824	**−0,095825**	**−0,095825**
	Std,	$2{,}49E-07$	$4{,}57E-14$	$2{,}83E-17$
g09	Best	681,305	681,866	**680,630**
	Mean	683,147	695,180	**680,630**
680,630	Worst	692,817	728,656	**680,630**
	Std,	$2{,}43E+00$	$1{,}12E+01$	$4{,}31E-13$
g10	Best	7102,727	7605,968	**7059,802**
	Mean	8619,988	10344,262	**7075,832**
7049,331	Worst	9977,791	13634,081	**7098,254**
	Std,	$1{,}40E+03$	$1{,}77E+03$	$8{,}54E+00$
g11	Best	0,750002	**0,750000**	**0,750000**
	Mean	0,750119	0,750050	**0,750000**
0,750000	Worst	0,750524	0,750387	**0,750000**
	Std,	$1{,}29E-04$	$8{,}31E-05$	$0{,}00E+00$
g12	Best	**−1,000000**	**−1,000000**	**−1,000000**
	Mean	**−1,000000**	**−1,000000**	**−1,000000**
−1,000000	Worst	−0,999997	**−1,000000**	**−1,000000**
	Std,	$5{,}73E-07$	$0{,}00E+00$	$0{,}00E+00$

TABLEAU 3.3: Résultats sur le *benchmark* des 12 fonctions, donnant pour chaque fonction la meilleure performance, la performance moyenne et la pire performance sur les 30 essais ("NF" signifie qu'aucune solution faisable n'a été trouvée).

Fonction	CBBO vs. CBBO-DE	CBBO vs. CBBO-PM	CBBO-DE vs. CBBO-PM
	t-test	t-test	t-test
g01	$1{,}09E+01\dagger$	$1{,}32E+01\dagger$	$4{,}51E-11$
g02	$2{,}51E+00\dagger$	$-2{,}77E+00\dagger$	$-3{,}71E+00\dagger$
g03	$5{,}99E+00\dagger$	$1{,}56E+00$	$-6{,}86E+00\dagger$
g04	$1{,}21E+01\dagger$	$5{,}09E+00\dagger$	$-9{,}47E+00\dagger$
g05	$6{,}12E+00\dagger$	$-1{,}50E+00$	$-4{,}94E+00\dagger$
g06	–	–	–
g07	$6{,}95E+00\dagger$	$-3{,}79E+00\dagger$	$-4{,}57E+00\dagger$
g08	$2{,}67E+00\dagger$	$2{,}67E+00\dagger$	$-1{,}02E+00$
g09	$5{,}67E+00\dagger$	$-5{,}75E+00\dagger$	$-7{,}11E+00\dagger$
g10	$6{,}02E+00\dagger$	$-4{,}19E+00\dagger$	$-1{,}01E+01\dagger$
g11	$5{,}06E+00\dagger$	$2{,}47E+00\dagger$	$-3{,}29E+00\dagger$
g12	$3{,}05E+00\dagger$	$3{,}05E+00\dagger$	$3{,}05E+00\dagger$

TABLEAU 3.4: Valeurs de la statistique t (en notation scientifique) d'un test t bilatéral avec 58 degrés de liberté et un seuil de signification de 0,05 († la différence entre les deux algorithmes est statistiquement significative)

3.3.3.1 Analyse statistique

Pour chaque fonction du *benchmark*, nous avons voulu vérifier si les différences entre les solutions trouvées par CBBO et les deux autres algorithmes étaient statistiquement significatives. À cet effet, nous avons utilisé le test t de Student. Le tableau 3.4 montre les résultats pour chaque paire d'algorithmes (CBBO et X). La valeur de la statistique t est calculée pour l'hypothèse nulle suivante : $\mathcal{H}_0$ = « *Il n'y a pas de différence significative entre la moyenne des solutions engendrées par CBBO et celle des solutions engendrées par X* ». Le tableau 3.4 montre les résultats de la comparaison statistique entre l'algorithme CBBO et les deux autres algorithmes (CBBO-DE et CBBO-PM), en utilisant le test t avec un nombre de degrés de liberté égal à $\nu = 30 + 30 - 2 = 58$ et un seuil de signification $\alpha = 0{,}05$.

Les résultats du test t peuvent être résumés comme suit :

1. **CBBO vs. CBBO-DE** : le tableau 3.4 montre que les valeurs de la statistique t sont toutes supérieures à la valeur critique pour les algorithmes CBBO et CBBO-DE pour toutes les fonctions de test. Nous pouvons rejeter l'hypothèse nulle $\mathcal{H}_0$ d'égalité des moyennes en faveur de l'hypothèse alternative et prétendre donc qu'il existe une différence statistique significative entre les deux algorithmes. Pour le problème $g06$, aucune solution faisable n'a été trouvée par l'algorithme CBBO, tandis que CBBO- DE a trouvé l'optimale.
2. **CBBO vs. CBBO-PM** : la valeur absolue de t est inférieure à la valeur critique pour deux cas de test ($g03$ et $g05$) et aucune solution faisable n'a été trouvée pour $g06$. Ces résultats viennent conforter l'hypothèse nulle selon laquelle il n'y aurait pas de différence

significative entre les deux algorithmes. Pour les 9 problèmes restants, nous pouvons rejeter l'hypothèse nulle d'égalité des moyennes en faveur de l'hypothèse alternative et en conclure que les moyennes obtenues par CBBO et CBBO-PM sont significativement différentes d'un point de vue statistique.

3. **CBBO-DE vs. CBBO-PM** : la valeur absolue de t est inférieure à la valeur critique pour 2 fonctions ($g01$ et $g08$). Par conséquent, nous ne pouvons pas rejeter l'hypothèse nulle. Pour les autres fonctions, le test t confirme que les moyennes obtenues par CBBO-DE et CBBO-PM sont significativement différentes. Nous pouvons ici rejeter avec confiance l'hypothèse nulle (les chances d'erreur étant de l'ordre de 5%).

Nous pouvons conclure que les algorithmes comparés sont statistiquement différents dans la plupart des cas. Il est également évident à partir des résultats que l'utilisation de la mutation de DE a plus d'effet sur l'algorithme CBBO que la méthode du pivot.

3.3.3.2 Effet de la stratégie de mutation

Pour étudier les effets des stratégies de mutation sur la performance de CBBO-DE, nous avons réalisé des tests avec différentes configurations. Chaque variante utilise un des schémas de mutation présenté dans la section 2.4.2.2. Les résultats sont visibles dans le tableau 3.5. Les valeurs en gras représentent les meilleurs résultats obtenus. En termes de meilleures valeurs de la fonction objectif (*Best*), CBBO-DE avec l'opérateur de mutation *DE/Current to best/1/bin* se trouve être le meilleur algorithme performant, car il obtient les valeurs optimales pour 10 fonctions de test sur un total de 12 fonctions (i.e., $g01$, $g02$, $g03$, $g05$, $g07$, $g08$, $g09$, $g10$, $g11$ and $g12$). Néanmoins, il a échoué dans le cas de la fonction $g06$, puisque aucune solution faisable n'a été trouvée durant les 30 essais réalisés. Pour l'algorithme CBBO-DE avec la stratégie de mutation *DE/Rand/1/bin*, l'optimum global est atteint dans 9 cas (i.e., $g01$, $g03$, $g04$, $g05$, $g06$, $g08$, $g09$, $g11$ and $g12$). Pour les autres fonctions de test, les résultats obtenus sont très proches de l'optimum global.

En termes de meilleurs valeurs moyennes de la fonction objectif (*Mean*), nous pouvons en déduire que CBBO-DE avec la stratégie de mutation *DE/Rand/1/bin* est nettement supérieur aux autres versions, car il produit un meilleur résultat dans 11 cas de test sur un total de 12. Par conséquent, nous retenons cette configuration pour la suite des tests avec d'autres méthodes de la littérature.

3.3.4 Comparaison avec d'autres algorithmes

Pour évaluer les performances de l'algorithme CBBO-DE, nous reprenons les expériences réalisées dans la section 3.3.3. Nous avons comparé les résultats de CBBO-DE à différents algo-

Fonction & Optimum	Statistiques	Schéma de mutation				
		DE/Rand/1/bin	DE/Best/1/bin	DE/Current to best/1/bin	DE/Best/2/bin	DE/Rand/2/bin
g01	Best	**−15,000**	**−15,000**	**−15,000**	**−15,000**	**−15,000**
−15,000	Mean	**−15,000**	**−15,000**	−14,993	−14,304	−13,849
	Worst	**−15,000**	**−15,000**	−14,935	−10,000	−9,415
g02	Best	−0,803557	−0,672202	**−0,803619**	−0,797885	−0,797856
−0,803619	Mean	**−0,802774**	−0,555723	−0,793968	−0,786006	−0,783795
	Worst	**−0,792576**	−0,464147	−0,777843	−0,752049	−0,754251
g03	Best	**−1,000**	**−1,000**	**−1,000**	−0,993	−0,996
−1,000	Mean	**−1,000**	**−1,000**	**−1,000**	−0,906	−0,924
	Worst	**−1,000**	**−1,000**	**−1,000**	−0,683	−0,611
g04	Best	**−30665,539**	**−30665,539**	−30651,808	−30657,320	−30663,652
−30665,539	Mean	**−30665,539**	**−30665,539**	−30604,354	−30572,855	−30588,958
	Worst	**−30665,539**	**−30665,539**	−30525,396	−30352,145	−30352,072
g05	Best	**5126,498**	**5126,498**	**5126,498**	5130,374	5133,023
5126,498	Mean	**5126,498**	5126,499	**5126,498**	5214,659	5222,979
	Worst	5126,499	5126,504	**5126,498**	5370,141	5538,816
g06	Best	**−6961,814**	*NF*	*NF*	*NF*	*NF*
6961,814	Mean	**−6961,814**	*NF*	*NF*	*NF*	*NF*
	Worst	**−6961,814**	*NF*	*NF*	*NF*	*NF*
g07	Best	24,326	24,313	**24,306**	29,217	27,583
24,306	Mean	24,345	24,419	**24,306**	60,841	60,356
	Worst	24,378	24,751	**24,306**	175,989	204,491
g08	Best	**−0,095825**	**−0,095825**	**−0,095825**	**−0,095825**	**−0,095825**
−0,095825	Mean	**−0,095825**	**−0,095825**	**−0,095825**	**−0,095825**	**−0,095825**
	Worst	**−0,095825**	**−0,095825**	**−0,095825**	**−0,095825**	**−0,095825**
g09	Best	**680,630**	**680,630**	**680,630**	683,693	683,629
680,630	Mean	**680,630**	**680,630**	**680,630**	725,404	717,678
	Worst	**680,630**	680,631	**680,630**	859,448	902,219
g10	Best	7059,803	7051,341	**7050,816**	7277,714	7732,944
7049,331	Mean	**7075,832**	7108,325	7078,041	9094,381	8956,845
	Worst	**7098,254**	7344,272	7199,885	11382,629	10541,834
g11	Best	**0,750000**	**0,750000**	**0,750000**	**0,750000**	**0,750000**
0,750000	Mean	**0,750000**	**0,750000**	**0,750000**	0,750684	0,750096
	Worst	**0,750000**	**0,750000**	**0,750000**	0,756737	0,750797
g12	Best	**−1,000000**	**−15,000**	**−1,000000**	**−1,000000**	**−1,000000**
−1,000000	Mean	**−1,000000**	**−1,000000**	**−1,000000**	−0,999429	−0,999623
	Worst	**−1,000000**	**−1,000000**	**−1,000000**	−0,994198	−0,994322

TABLEAU 3.5: Comparaison des résultats de CBBO-DE avec différents schémas de mutation («NF» signifie qu'aucune solution réalisable n'a été trouvée).

Algorithmes	Nombre d'évaluations	Nombre d'essais	Taille de la population
CBBO-DE	350 000	30	100
CRGA [Amirjanov, 2006]	350 000	30	100
SR [Runarsson & Yao, 2000]	350 000	30	200
PSO [Pulido & Coello, 2004]	340 000	30	40
CULDE [Becerra & Coello, 2005]	100 100	30	100
SMDE [Mezura-Montes & Coello, 2005]	240 000	30	300

TABLEAU 3.6: Principaux paramètres des algorithmes en comparaison

rithmes s'étant confrontés au même *benchmark*. Nous avons alors récupéré les résultats de cinq méthodes différentes. Nous ne détaillerons pas ici chacune de ces méthodes, nous engageons donc le lecteur à se référer aux articles cités pour de plus amples informations. Les différents algorithmes sont : *Changing Range Genetic Algorithm* (CRGA) [Amirjanov, 2006], *Stochastic Ranking approach* (SR) [Runarsson & Yao, 2000], *Particle Swarm Optimization algorithm* (PSO) [Pulido & Coello, 2004], *Cultured Differential Evolution approach* (CULDE) [Becerra & Coello, 2005] et *Simple Multimembered Evolution Strategy* (SMDE) [Mezura-Montes & Coello, 2005]. Le tableau 3.6 montre pour chaque algorithme, le nombre d'évaluations requises, le nombre d'essais réalisés et la taille de la population. Le reste des paramètres est indiqué dans les références respectives des algorithmes. Les paramètres spécifiques à CBBO-DE gardent les mêmes valeurs que celles décrites dans le tableau 3.2. Les résultats de ces algorithmes ont été consignés dans le tableau 3.7. Pour plus de lisibilité, nous avons mis en gras les algorithmes qui obtenaient les meilleurs résultats pour chaque fonction.

Nous pouvons voir dans ce tableau 3.7 que les résultats donnés par CBBO-DE sont d'une qualité très satisfaisante. Notre algorithme obtient les meilleurs résultats (*Best*) avec 9 fonctions sur 12 résolues parfaitement. Il est cependant devancé par l'algorithme RS qui, lui, obtient les meilleurs résultats pour 10 fonctions, mais ne converge vers l'optimum que 8 fois sur 12, contre 9 fois sur 12 pour CBBO-DE. En revanche, notre algorithme semble obtenir les meilleurs résultats (*Mean* et *Worst*), comparé à toutes les autres approches (10 cas sur 12 pour CBBO-DE, 6 cas pour SR et 8 cas pour CULDE).

3.4 Conclusion

Dans ce chapitre, nous avons présenté trois nouveaux algorithmes : CBBO (Constrained BBO), CBBO-DE (Constrained BBO avec l'opérateur de mutation de DE) et CBBO-PM (Constrained BBO avec la méthode *Pivot-2*). Ces algorithmes sont issus des adaptations de BBO pour la prise en compte des contraintes. Les approches proposées disposent d'une stratégie de sélection basée sur le principe de la supériorité des solutions faisables, d'une méthode de pé-

Fonction & Optimum	Statistiques	Algorithme					
		CBBO-DE	CRGA	SR	PSO	CULDE	SMDE
g01	Best	**−15.000**	−14.9977	**−15.000**	**−15.000**	**−15.000**	**−15.000**
	Mean	**−15.000**	−14.985	**−15.000**	**−15.000**	**−15.000**	**−15.000**
−15.000	Worst	**−15.000**	−14.9467	**−15.000**	**−15.000**	**−15.000**	**−15.000**
	Std.	$8.21E-14$	NA	$0.0E+00$	NA	$0.2E-06$	$0.0E+00$
g02	Best	−0.803557	−0.802959	−0.803515	−0.803432	**−0.803619**	0.803601
	Mean	**−0.802774**	0.764494	−0.781975	0.790406	0.724886	0.785238
−0.803619	Worst	**−0.792576**	−0.722109	−0.726288	0.750393	0.590908	0.751322
	Std.	$2.72E-03$	NA	$2.0E-02$	NA	$7.01E-03$	$1.67E-02$
g03	Best	**−1.000**	0.9997	**−1.000**	1.004720	−0995413	**−1.000**
	Mean	**−1.000**	0.9972	**−1.000**	1.003814	−0.788635	**−1.000**
−1.000	Worst	**−1.000**	0.9931	**−1.000**	1.002490	−0639920	**−1.000**
	Std.	$6.04E-16$	NA	$1.9E-04$	NA	$1.11E-02$	$2.09E-04$
g04	Best	**−30665.539**	−30665.520	**−30665.539**	−30665.500	**−30665.539**	**−30665.539**
	Mean	**−30665.539**	−30664.398	**−30665.539**	−30665.500	**−30665.539**	**−30665.539**
−30665.539	Worst	**−30665.539**	−30660.313	**−30665.539**	−30665.500	**−30665.539**	**−30665.539**
	Std.	$1.67E-11$	NA	$2.0E-05$	NA	$0.0E+00$	$0.0E+00$
g05	Best	**5126.498**	5126.500	5126.497	5126.640	5126.571	5126.599
	Mean	**5126.498**	5507.041	5128.881	5461.081	5207.411	5174.492
5126.498	Worst	**5126.498**	6112.075	5142.472	6104.750	5327.391	5304.167
	Std.	$2.23E-04$	NA	$3.5E+00$	NA	$69.22E+00$	$50.06E+00$
g06	Best	**−6961.814**	−6956.251	**−6961.814**	−6961.810	**−6961.814**	**−6961.814**
	Mean	**−6961.814**	−6740.288	−6875.940	−6961.810	**−6961.814**	−6961.284
6961.814	Worst	**−6961.814**	−6077.123	−6350.262	−6961.810	**−6961.814**	−6952.482
	Std.	$4.55E-12$	NA	$1.6E+02$	NA	$0.0E+00$	$1.85E+00$
g07	Best	24.326	24.882	**24.307**	24.351	**24.307**	24.327
	Mean	24.345	25.746	24.374	25.356	**24.306**	24.475
24.306	Worst	24.378	27.381	24.642	27.317	**24.306**	24.843
	Std.	$1.33E-02$	NA	$6.6E-02$	NA	$1.00E-06$	$1.32E-01$
g08	Best	**−0.095825**	**−0.095825**	**−0.095825**	**−0.095825**	**−0.095825**	**−0.095825**
	Mean	**−0.095825**	−0.095819	**−0.095825**	**−0.095825**	**−0.095825**	**−0.095825**
−0.095825	Worst	**−0.095825**	−0.095808	**−0.095825**	**−0.095825**	**−0.095825**	**−0.095825**
	Std.	$2.83E-17$	NA	$2.6E-17$	NA	$0.0E+00$	$0.0E+00$
g09	Best	**680.630**	680.726	**680.630**	680.638	**680.630**	680.632
	Mean	**680.630**	681.347	680.656	680.852	**680.630**	680.643
680.630	Worst	**680.630**	682.965	680.763	681.553	**680.630**	680.719
	Std.	$4.31E-13$	NA	$3.4E-2$	NA	$0.0E+00$	$1.55E-02$
g10	Best	7059.802	7114.743	**7054.316**	7057.590	7049.248	7051.903
	Mean	7075.832	8785.149	7559.192	7560.048	**7049.248**	7253.047
7049.331	Worst	7098.254	10826.09	8835.655	8104.310	**7049.248**	7638.366
	Std.	$8.54E+00$	NA	$5.3E+02$	NA	$1.67E-04$	$136.02E+00$
g11	Best	**0.750**	**0.750**	**0.750**	**0.750**	**0.750**	**0.750**
	Mean	**0.750**	0.752	**0.750**	**0.750**	0.758	**0.750**
0.750000	Worst	**0.750**	0.757	**0.750**	0.753	0.796	**0.750**
	Std.	$0.00E+00$	NA	$8.0E-05$	NA	$1.71E-02$	$1.52E-04$
g12	Best	**−1.000000**	**−1.000000**	**−1.000000**	**−1.000000**	**−1.000000**	**−1.000000**
	Mean	**−1.000000**	**−1.000000**	**−1.000000**	**−1.000000**	**−1.000000**	**−1.000000**
−1.000000	Worst	**−1.000000**	**−1.000000**	**−1.000000**	**−1.000000**	**−1.000000**	**−1.000000**
	Std.	$0.00E+00$	NA	$0.00E+00$	NA	$0.00E+00$	$0.00E+00$

TABLEAU 3.7: Résultats sur le *benchmark* des 12 fonctions, donnant pour chaque fonction la meilleure performance, la performance moyenne et la pire performance sur les 30 essais ("NA" signifie n'est pas disponible).

nalisation qui permet à la fonction objectif de prendre en compte les violations des contraintes en pénalisant les individus concernés et d'une stratégie de classement stochastique. Elles diffèrent les unes des autres par la manière de faire évoluer la population à travers des opérateurs de variation propres à chaque méthode.

Ces algorithmes ont été testés sur différentes fonctions objectifs avec des contraintes d'égalités et d'inégalité, afin d'évaluer leurs performances et de comparer leurs caractéristiques. CBBO-DE obtenant des résultats de meilleure qualité que CBBO et CBBO-PM, il a été retenu pour la comparaison avec d'autres méthodes de la littérature destinés aux problèmes contraints. Les résultats obtenus sont très satisfaisants.

Les pistes d'amélioration de CBBO impliquent de tester d'autres approches de prise en compte des contraintes et de continuer à explorer la possibilité de développer de nouvelles approches. Une idée serait, par exemple, l'introduction d'un mécanisme de contrôle de la diversité.

Chapitre Quatre

Application des CBBOs pour l'allocation optimale de puissance dans les réseaux de capteurs sans fil

4.1 Introduction et positionnement du problème

Depuis quelques décennies, le besoin d'observer et de contrôler des phénomènes physiques tels que la température, la pression ou encore la luminosité est essentiel pour de nombreuses applications industrielles et scientifiques. Les récentes avancées technologiques dans les domaines de la microélectronique, de la micromécanique, et des technologies de communication sans fil ont permis de mettre en évidence un nouveau champ de recherche multidisciplinaire, celui des réseaux de capteurs sans fil (RCSF) (WSN : *Wireless Sensor Network*). Les RCSFs fournissent une solution de collecte distribuée de données dans des environnements physiques parfois difficiles d'accès. Initialement destinés à des applications militaires, leurs domaines d'application couvrent désormais un large spectre allant notamment de la collecte de données environnementales à la surveillance d'infrastructures industrielles en passant par l'aide aux personnes [Akyildiz et al., 2002].

Un RCSF est un réseau ad-hoc [1] constitué d'un grand nombre de nœuds qui sont des microcapteurs capables de détecter, mesurer et de transmettre des données physiques liées à leur environnement d'une manière autonome. La position de ces nœuds n'est pas obligatoirement prédéterminée. Ils peuvent être aléatoirement dispersés dans une zone géographique, appelée « champ de captage » correspondant au terrain d'intérêt pour le phénomène capté.

La consommation d'énergie dans les RCSFs, fortement affectée par les communications entre les nœuds, est un problème important. En effet, la dépendance à l'égard d'une énergie limitée fournie par une batterie constitue souvent un obstacle majeur. La durée de vie des capteurs est par conséquent égale à la durée de vie de leur batterie. Il est donc largement reconnu que la réduction de la consommation énergétique, ou du moins sa rationalisation, est une question

1. Un réseau ad-hoc est un réseau sans fil capable de s'organiser sans la présence d'infrastructure réseau (i.e., routeurs). Les réseaux de capteurs doivent être ad-hoc, tout d'abord dans un souci de simplicité d'installation, mais aussi et surtout dans le souci de permettre au réseau de rester opérationnel même après des défaillances ponctuelles de nœuds.

incontournable dans la conception de ce type de réseaux.

Le domaine de la détection, au cœur d'un grand nombre d'applications, a été largement étudié et a donné lieu à plusieurs approches. On distingue la *détection centralisée* dans laquelle l'ensemble des capteurs, déployés autour ou dans une zone à observer, font des mesures périodiques et communiquent directement leurs observations à une station de traitement appelée communément le puits (*sink*) ou le *centre de fusion* qui prend la décision finale. Cette approche nécessite souvent une grande capacité de transport d'information.

Une autre approche, qui résulte de la volonté de réduire la quantité d'information transmise au sein du système, est la *détection décentralisée* (également appelée *détection distribuée*)[Tenny & Sandell, 1981]. Dans un système de détection décentralisée, chaque capteur effectue un traitement préliminaire des données d'une manière répartie et transmet une décision locale au centre de fusion. À son tour, le centre de fusion traite les données reçues et les combine afin de prendre la décision finale. La principale différence entre cette approche et le système de décision centralisé classique est que le centre de fusion n'a pas accès à l'observation brute réalisée au niveau de chaque capteur. Évidemment, un système distribué est sous-optimal par rapport à un système centralisé, dans lequel le centre de fusion a accès aux observations locales de tous les capteurs sans distorsion. Toutefois, les approches distribuées offrent la possibilité de réductions drastiques des besoins de communication et d'énergie nécessaire pour obtenir une estimation précise, au détriment d'une certaine dégradation de performances [Tsitsiklis, 1993].

En raison des fortes contraintes en ressources énergétiques, en mémoire, en bande passante, et en complexité de calcul, le problème standard dans un système de détection distribuée est de déterminer les règles de décision locales au niveau des nœuds capteurs et la règle de décision au centre de fusion qui optimisent les performances de détection par rapport à un certain critère de performance donné [2]. La prise de décision consiste alors à poser un problème d'optimisation, qui consiste à minimiser la puissance sans que la probabilité d'erreur au centre de fusion ne dépasse une valeur limite. Nous sommes ainsi face à un problème d'optimisation sous contrainte. Nous nous proposons de traiter ce problème par une technique d'optimisation stochastique.

Dans ce chapitre, le problème de l'allocation optimale de puissance dans un système à plusieurs capteurs et un centre de fusion sera abordé, dans le contexte d'un canal Rayleigh de transmission à évanouissement. Tout d'abord, nous présenterons le problème de détection décentralisée. Le critère de qualité que nous proposons d'utiliser est défini comme la probabilité

2. Les règles de décision utilisées reposent souvent sur le risque moyen, le maximum de gain, le minimum de risque, tous issus de l'approche bayésienne, mais aussi sur des critères tels que celui de Neyman-Pearson [Neyman & Pearson, 1933], consistant à maximiser la probabilité de détection pour une probabilité de fausse alarme donnée. Cela suppose implicitement que la fausse alarme est considérée comme l'erreur la plus grave, ce qui n'est pas toujours le cas suivant les applications (par exemple dans le cas du déminage humanitaire, c'est la non-détection qui est l'erreur la plus grave).

d'erreur de détection au centre de fusion, et repose sur les probabilités de fausse alarme et de détection.

Une résolution analytique du problème d'allocation de puissance par la méthode des multiplicateurs de Lagrange est présentée dans le cas où les observations des capteurs sont indépendantes et identiquement distribuées (i.i.d.). Nous nous intéressons également dans cette étude au cas des observations spatialement corrélées, où il est difficile de trouver une solution analytique. Une résolution à l'aide d'algorithmes d'optimisation stochastiques est donc adoptée [Boussaïd et al., 2011b, 2013b]. Finalement, nous étudions les performances de trois versions de BBO, et nous observons numériquement les gains obtenus par les schémas d'allocation de puissance proposés par rapport à ceux obtenus avec les algorithmes CDE, CGA et CPSO. Ces algorithmes sont des adaptations, au problème d'optimisation avec contraintes, des algorithmes DE, GA et PSO respectivement, que nous avons développé pour les besoins de notre étude.

4.2 Présentation du problème de détection

Le problème le plus simple relevant de la théorie de la décision est le problème qui consiste à devoir choisir entre plusieurs hypothèses possibles (en général, il y a $M \geq 2$ hypothèses $\mathcal{H}_0, \ldots \mathcal{H}_{M-1}$) qui permettent chacune d'expliquer ou d'interpréter des données observées. Il s'agit donc de décider à partir d'un ensemble d'observations collectées, quelle est l'hypothèse qui décrit le mieux les observations mesurées. Dans le cadre de la surveillance d'un système par exemple, chaque hypothèse est généralement associée à un état du système et il s'agit d'identifier cet état. La qualité du test associé est alors naturellement définie par le nombre d'erreurs de décision, ce qui, en statistique, se traduit par des probabilités d'erreurs. Nous appelons probabilité d'erreur, la probabilité de rejeter $\mathcal{H}_i$ alors que $\mathcal{H}_i$ est vraie.

Les observations correspondent le plus souvent à des mesures effectuées par des capteurs qui surveillent le système. Sous chacune des hypothèses, les données collectées sont supposées aléatoires car elle dépendent d'éléments qui sont aléatoires par nature. A partir des mesures dans l'espace d'observation, nous devons finalement décider quelle est l'hypothèse la plus vraisemblable, au moyen d'une règle qui assigne une hypothèse à chaque point de l'espace d'observation. De façon plus globale, la *règle de décision* partitionne l'espace d'observation en M régions, chacune associée à une hypothèse.

Deux grands types de problème coexistent dans la littérature : la décision entre deux hypothèses uniquement et la décision entre plusieurs hypothèses (au moins trois) [Levy, 2008]. Décider entre deux hypothèses revient à ranger les observations mesurées en deux classes ; chaque classe correspond à une situation donnée. Une hypothèse est qualifiée d'hypothèse de

base et l'autre hypothèse est qualifiée d'alternative. Par exemple, en radar, l'hypothèse de base correspond à l'absence d'échos dans les mesures radar et l'hypothèse alternative correspond à la présence d'un ou plusieurs objets dans les mesures radar. Nous parlons alors d'un *problème de détection* dont la problématique majeure est de réussir à isoler dans le signal reçu les cibles d'intérêt des perturbations causées par l'environnement.

Pour illustrer les objectifs de la théorie de la détection, considérons l'exemple d'un système de communications numériques destiné à transmettre des informations d'un émetteur à un ou plusieurs récepteurs. L'émetteur envoie, au travers d'un canal de transmission imparfait, un nombre fini de symboles $S_0, S_1, \ldots, S_{M-1}$ [3]. Les messages porteurs d'information générés par la source sont transmis vers le destinataire à l'aide d'un signal. Ce signal peut prendre une infinité de valeurs différentes et subit lors de sa propagation des modifications ou des altérations pouvant conduire à des erreurs d'interprétations du signal recueilli par le récepteur. Ces modifications sont le plus souvent dues à la présence de bruit, à des atténuations, à des effets d'interférences ou à un certain nombre de défauts inhérents au canal de transmission choisi. À la réception, nous faisons des hypothèses au sujet des symboles fournis par la source. Chaque hypothèse $\mathcal{H}_0, \ldots, \mathcal{H}_{M-1}$ est relative à un seul symbole $S_0, S_1, \ldots, S_{M-1}$. Sur la base du signal reçu à la sortie du canal, le récepteur prend des décisions afin d'établir quelle hypothèse, donc quel symbole généré par la source, correspond au signal reçu. La tâche du détecteur est donc de retrouver le symbole émis à partir d'observations noyées dans un bruit, ou de prendre toute autre décision sur la nature du signal observé, en faisant un minimum d'erreurs de décision.

Dans le cas de tests d'hypothèses binaires, la tâche du détecteur est de décider laquelle des deux hypothèses possibles, $\mathcal{H}_0$ (*hypothèse nulle* ou *hypothèse de bruit seul*) et $\mathcal{H}_1$ (*hypothèse de présence de signal*) est vraie. Classiquement, les critères de Bayes et de Neyman-Pearson sont utilisés dans la littérature. La formulation Bayesienne de ce type de problème est basée sur la connaissance, pour chaque hypothèse $\mathcal{H}_i$ $(i = 0, 1)$, de la probabilité *a priori* pour que cette hypothèse se réalise. L'approche Bayesienne associe, à chaque situation possible du système de décision, un coût (équivalent à une pénalisation ou une récompense) et une fonction *risque moyen* (ou probabilité d'erreur).

Un critère de performance possible dans la formulation de Bayes est de trouver un détecteur qui minimise la probabilité d'erreur $\mathcal{P}_E$. La réalisation d'un test sera évaluée en fonction de trois quantités : la probabilité de détection $\mathcal{P}_D$, la probabilité de non-détection ou détection manquée $(\mathcal{P}_M = 1 - \mathcal{P}_D)$ et la probabilité de fausse alarme $\mathcal{P}_F$. La fausse alarme, parfois qualifiée aussi de niveau du test, consiste à décider en faveur de l'hypothèse alternative $\mathcal{H}_1$ alors que c'est

3. Ces symboles peuvent être représentés par les symboles 0 et 1 dans la cas des transmissions codées, la présence ou l'absence d'un avion dans le cas du radar, la présence ou l'absence d'un navire dans le cas du sonar, etc.

l'hypothèse de base $\mathcal{H}_0$ qui est correcte. La détection manquée consiste à décider en faveur de l'hypothèse de base alors que l'hypothèse alternative est vraie.

Dans les applications pour lesquelles la probabilité *a priori*, d'occurrence de chaque hypothèse, est connue, l'approche Bayésienne est une excellente solution au problème de décision. Ce n'est cependant pas forcément le cas pour toutes les applications et une approche alternative serait l'utilisation du critère de Neyman-Pearson [Neyman & Pearson, 1933]. Ce critère vise à maximiser (minimiser) la probabilité de détection $\mathcal{P}_D$ (la probabilité de non-détection $\mathcal{P}_M$), sachant que la probabilité $\mathcal{P}_F$ de fausse alarme du système global doit être inférieure ou égale à une valeur fixée α. En pratique, ce taux est spécifié par l'utilisateur et choisi en fonction de l'application.

Par souci de simplicité, nous considérons la formulation Bayésienne basée sur la minimisation d'un unique critère (la probabilité d'erreur au centre de fusion) plutôt que la formulation de Neyman-Pearson qui cherche des tests optimaux suivant deux critères simultanément (maximiser la probabilité de détection, sous contrainte d'une probabilité de fausse alarme bornée) [Kay, 1998; Poor, 1998; Trees, 1998].

La section suivante présente une formulation précise des règles de décision au niveau de chaque capteur et de la règle de fusion optimale. Les décisions locales sont transmises via un canal de transmission sujet à des perturbations de type évanouissement (ou *fading*) de Rayleigh. Ces perturbations introduisent des erreurs dans la transmission des données, pouvant conduire à des erreurs d'interprétations du signal reçu. Sur la base des données reçues, le centre de fusion résout un problème de décision binaire et statue sur l'une des hypothèses possibles [Tsitsiklis, 1993]. Cependant, une question qui se pose ici est la suivante : « *comment combiner efficacement, et de manière optimale, les décisions locales des nœuds capteurs dans le réseau afin de s'assurer que la décision finale soit atteinte avec un haut degré de fiabilité ?* » Dans cette étude, nous nous intéresserons au problème de décision binaire en adoptant l'approche Bayesienne. La règle de fusion avec observations i.i.d. et corrélées a été étudiée. L'approche de détection décentralisée développée dans ce chapitre est inspiré des travaux de Wimalajeewa et al. [Wimalajeewa & Jayaweera, 2008].

4.3 Formulation du problème

Considérons le système de détection distribué de la Figure 4.1, dont la structure de base est composée d'un ensemble de L capteurs $(S_1, \ldots, S_L)$ identiques. Ce réseau comporte, de plus, un centre de fusion qui rassemble les données des capteurs et prend la décision finale. Chaque capteur effectue une décision locale, basée sur ses propres observations, et la transmet

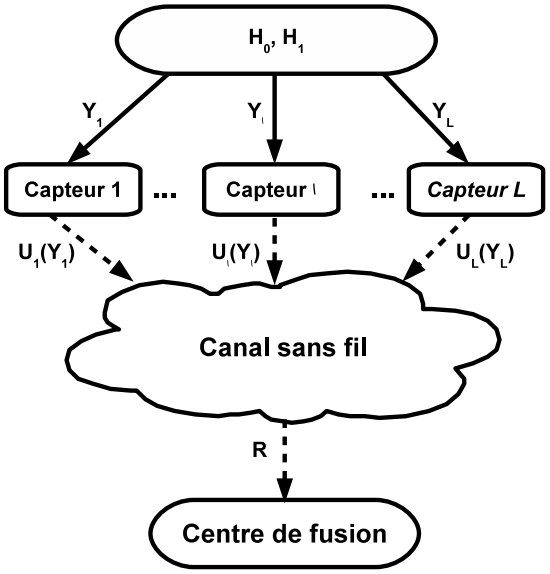


FIGURE 4.1: Problème de détection décentralisée – Test d'hypothèse binaire

au centre de fusion de données où une décision globale est obtenue. Chaque capteur observe un seul composant du vecteur $\mathbf{X} = [X_1, \ldots, X_L]^T$ du signal source à détecter.

Le système étudié a pour mission de résoudre un problème de détection, c'est-à-dire un problème de choix entre deux hypothèse $\mathcal{H}_0$ (bruit seul) et $\mathcal{H}_1$ (signal et bruit) à partir d'observations collectées sur l'ensemble des L source d'information. Les lois de probabilités *a priori* π_j $(j = 0, 1)$ sont associées à chaque hypothèse, et nous notons $\pi_0 = \mathcal{P}(\mathcal{H}_0)$ et $\pi_1 = \mathcal{P}(\mathcal{H}_1)$.

Considérons à présent le problème de détection, dans un vecteur d'observation $\mathbf{Y}$ de dimension L, d'un signal constant X corrompu par un bruit additif gaussien. Ce problème se formalise généralement par un test d'hypothèses binaires. Ainsi, l'observation locale Y_ℓ, obtenu au niveau du capteur ℓ, sous chacune des deux hypothèses de test, est donnée par :

$$\begin{aligned} \mathcal{H}_0 &: Y_\ell = V_\ell; \quad \ell = 1,2,\ldots,L \\ \mathcal{H}_1 &: Y_\ell = X_\ell + V_\ell; \quad \ell = 1,2,\ldots,L \end{aligned} \tag{4.3.1}$$

Nous nous intéressons dans ce travail au problème de la détection d'un signal déterministe ($X_\ell = m$ pour $\ell = 1, 2, \ldots, L$, où $m > 0$). Les mesures sont polluées par un bruit additif gaussien, noté V_ℓ, de moyenne nulle et de variance σ_v^2.

Les observations peuvent s'écrire sous forme vectorielle : $\mathbf{Y} = \mathbf{X}+\mathbf{V}$, où $\mathbf{V} = [V_1, V_2, \ldots, V_L]^T$ est un vecteur de bruit gaussien de moyenne zéro et de matrice de covariance $\mathbf{\Sigma_v}$ (dans la cas général, la matrice de covariance $\mathbf{\Sigma_v}$ n'est pas diagonale sauf en présence de bruit d'observation dont les composantes sont indépendantes et identiquement distribuées (i.i.d.)),

$\mathbf{X} = [X_1, X_2, \ldots, X_L]^T$ représente le signal source à détecter, où le symbole T désigne l'opérateur transposé d'un vecteur.

On désigne par $\gamma_0 = m^2/\sigma_v^2$ le rapport signal à bruit (*signal-to-noise ratio*, SNR en anglais) qui est défini comme le rapport de la puissance du signal utile reçu sur la puissance de bruit.

4.3.1 Règles de décisions locales

Chaque capteur effectue une décision locale $U_\ell(Y_\ell)$, basée sur ses propres observations. Les décisions locales sont ensuite transmises au centre de fusion de données qui les combine de façon à obtenir la décision finale. Considérons, par soucis de simplification, le cas où les capteurs transmettent au centre de fusion une version amplifiée de leur observation. Ce type de capteur se révèle efficace lorsque les observations au niveau des capteurs sont corrompus par un bruit additif [Chamberland & Veeravalli, 2004]. Chaque capteur agit donc comme un *analog relay amplifier* avec une fonction de transmission donnée par :

$$U_\ell(Y_\ell) = G_\ell Y_\ell; \quad \ell = 1,2,\ldots,L \tag{4.3.2}$$

où G_ℓ est le gain de l'amplificateur au nœud ℓ.

4.3.2 Transmission des décisions locales

Les décisions locales U_ℓ ($\ell = 1,2,\ldots,L$) réalisées au niveau de chaque capteur sont acheminées au centre de fusion de données via un canal de communication sans fil. La transmission est sujette à une atténuation et à un effet d'évanouissement (*fading*). Nous supposons que le centre de fusion reçoit de chaque capteur ℓ, $\ell = 1,2,\ldots,L$, une information notée R_ℓ. Le signal reçu, sous chacune des hypothèses $\mathcal{H}_j$, $j = 0, 1$, est donné par :

$$\begin{aligned} &\mathcal{H}_0 : R_\ell = N_\ell; \quad \ell = 1,2,\ldots,L \\ &\mathcal{H}_1 : R_\ell = H_\ell G_\ell X_\ell + N_\ell; \quad \ell = 1,2,\ldots,L \end{aligned} \tag{4.3.3}$$

où H_ℓ représente le cœfficient de *fading* entre la source et la destination, $N_\ell = H_\ell G_\ell V_\ell + W_\ell$, où les W_ℓ, $1 \leq \ell \leq L$ sont des séquences aléatoires (i.i.d.) suivant une loi normale de moyenne nulle et de variance égale à σ_w^2. N_ℓ est un bruit gaussien au centre de fusion de moyenne zéro et de matrice de covariance $\mathbf{\Sigma}_n$, donnée par :

$$\mathbf{\Sigma}_n = H_\ell G_\ell \mathbf{\Sigma}_v G_\ell H_\ell + \mathbf{\Sigma}_w \tag{4.3.4}$$

où $\mathbf{\Sigma}_w$ représente la matrice de covariance du bruit du récepteur.

Sous l'hypothèse $\mathcal{H}_0$, le signal reçu par le centre de fusion est supposé contenir uniquement le bruit. Sous l'hypothèse $\mathcal{H}_1$, le signal reçu est supposé contenir les décisions partielles provenant des capteurs mais noyées parmi le bruit. Le système (4.3.3) peut s'écrire sous la forme vectorielle : $\mathbf{R} = \mathbf{AX} + \mathbf{N}$ où $\mathbf{A} = diag(H_1G_1, H_2G_2, \ldots, H_LG_L)$, $\mathbf{R} = [R_1, R_2, \ldots, R_L]^T$ et correspond à l'information reçue au centre de fusion, $\mathbf{X} = [X_1, X_2, \ldots, X_L]^T$ est le vecteur du signal observé et $\mathbf{N} = [N_1, N_2, \ldots, N_L]^T$ correspond au bruit. L'expression précédente de la matrice de covariance $\mathbf{\Sigma}_n$ du bruit au centre de fusion (équation 4.3.4) peut alors être réécrite comme :

$$\mathbf{\Sigma}_n = \mathbf{A}^T\mathbf{\Sigma}_v\mathbf{A} + \Sigma_w \tag{4.3.5}$$

4.3.3 Fusion des décisions

Le centre de fusion dispose de la suite d'observations $\mathbf{R} = [R_1, \ldots, R_L]^T$, suivant une loi normale :

$$\begin{aligned} \mathcal{H}_0 &: \mathbf{R} \sim \mathcal{N}(0, \mathbf{\Sigma}_n) \\ \mathcal{H}_1 &: \mathbf{R} \sim \mathcal{N}(\mathbf{AM}, \mathbf{\Sigma}_n) \end{aligned} \tag{4.3.6}$$

où $\mathcal{N}$ dénote une distribution gaussienne, $\mathbf{M} = m\mathbf{e}$ et $\mathbf{e}$ est le vecteur de taille L et dont toutes les composantes sont égales à 1. L'objectif du système consiste donc à décider lequel des deux signaux possibles est présent. Nous avons établi dans la section 4.2 que le test d'hypothèse binaire conduit à comparer le rapport de vraisemblance à un seuil. Ce seuil est fonction des lois de probabilités *a priori* sous chacune des hypothèses. Il a été également souligné que dans l'approche bayesienne, les probabilités de réalisation de chacune des hypothèses sont supposées connues. La décision optimale (qui minimise le critère de Bayes) est alors obtenue en comparant le logarithme du rapport de vraisemblance (LLR : *Log Likelihood Ratio*) du vecteur d'observation, noté $T(\mathbf{R})$, à un seuil noté τ [Poor, 1998]. Ce seuil s'exprime en fonction des probabilités *a priori*, π_0 et π_1. Considérons que le seuil de détection τ soit égal à π_0/π_1. Le LLR $T(\mathbf{R})$ pour le problème de détection peut s'écrire sous la forme suivante :

$$T(\mathbf{R}) = m\mathbf{e}^T\mathbf{A}\mathbf{\Sigma}_n^{-1}\mathbf{R} - \frac{1}{2}m^2\mathbf{e}^T\mathbf{A}\mathbf{\Sigma}_n^{-1}\mathbf{Ae} \tag{4.3.7}$$

On formalise la règle de décision Bayesienne optimale de la manière suivante :

$$\delta(\mathbf{R}) = \begin{cases} 1 & \text{if } T(\mathbf{R}) \geq \ln\tau \\ 0 & \text{if } T(\mathbf{R}) < \ln\tau \end{cases} \tag{4.3.8}$$

où ln représente la fonction logarithme néperien.

Le rapport de vraisemblance logarithmique a la distribution de probabilité suivante, sous chacune des hypothèses :

$$\begin{array}{l} \mathcal{H}_0 : T(\mathbf{R}) \sim \mathcal{N}(-\frac{1}{2}m^2\mathbf{e}^T\mathbf{A}\mathbf{\Sigma}_n^{-1}\mathbf{A}\mathbf{e}, m^2\mathbf{e}^T\mathbf{A}\mathbf{\Sigma}_n^{-1}\mathbf{A}\mathbf{e}) \\ \mathcal{H}_1 : T(\mathbf{R}) \sim \mathcal{N}(\frac{1}{2}m^2\mathbf{e}^T\mathbf{A}\mathbf{\Sigma}_n^{-1}\mathbf{A}\mathbf{e}, m^2\mathbf{e}^T\mathbf{A}\mathbf{\Sigma}_n^{-1}\mathbf{A}\mathbf{e}) \end{array} . \tag{4.3.9}$$

Si l'on se place dans le cas où les hypothèses $\mathcal{H}_0$ et $\mathcal{H}_1$ sont équiprobables, autrement dit : $\pi_0 = \pi_1 = 1/2$, le seuil de décision optimale au niveau du centre de fusion vaut donc 1 ($\tau = \pi_0/\pi_1 = 1$). De plus, nous considérons que le critère à optimiser est la minimisation de la probabilité d'erreur. Il convient donc d'exprimer cette dernière. En notant $\mathcal{P}_E$ la probabilité d'erreur, il suffit de considérer qu'il y a erreur lorsque le centre de fusion choisit l'hypothèse $\mathcal{H}_1$ lorsque $\mathcal{H}_0$ est vraie. La probabilité d'erreur $\mathcal{P}_E$ s'exprime donc en fonction des probabilités de fausse alarme $\mathcal{P}_F$ et de détection $\mathcal{P}_D$. Sa formalisation donne alors :

$$\begin{array}{rl} \mathcal{P}_E & = \mathcal{P}_F\pi_0 + (1-\mathcal{P}_D)\pi_1 \\ & = \mathcal{Q}\left(\frac{1}{2}\sqrt{m^2\mathbf{e}^T\mathbf{A}\mathbf{\Sigma}_n^{-1}\mathbf{A}\mathbf{e}}\right) \end{array} . \tag{4.3.10}$$

où π_j est la probabilité *a priori* associée à l'hypothèse $\mathcal{H}_j$, $j = 0, 1$, $\mathcal{P}_F$ est la probabilité de fausse alarme, $\mathcal{P}_D$ est la probabilité de détection, et $\mathcal{Q}(.)$ est la fonction de distribution cumulative complémentaire normale :

$$\mathcal{Q}(x) = \int_x^\infty \frac{1}{\sqrt{2\pi}} e^{\frac{-\xi^2}{2}} d\xi \tag{4.3.11}$$

4.4 L'allocation optimale de puissance

Le vrai défi critique dans les RCSFs est l'énergie, car les capteurs sont souvent dotés de batteries non rechargeables. Ainsi, l'objectif principal dans ces réseaux est de minimiser la consommation d'énergie tout en assurant que le réseau effectue sa tâche dans des meilleures conditions. Par conséquent, un schéma d'allocation de puissance optimale s'impose. Nous abordons dans cette section le problème de la répartition optimale de puissance entre les capteurs dans un système de détection distribué. Comme il a été discuté dans l'introduction, la nature distribuée des observations couplées avec les contraintes de bande passante et de ressources énergétiques nécessite un moyen de combiner les observations locales des capteurs tout en gardant l'erreur de fusion au dessous d'une valeur limite.

On s'intéresse donc au problème d'optimisation suivant :

$$\begin{cases} \min \sum_{\ell=1}^{L} G_\ell^2 & \\ \text{s.c.} & \mathcal{P}_E = Q\left(\frac{1}{2}\sqrt{m^2\mathbf{e}^T\mathbf{A}\mathbf{\Sigma}_n^{-1}\mathbf{A}\mathbf{e}}\right) \leq \epsilon \\ & G_\ell \geq 0; \quad \ell = 1, 2, \ldots, L \end{cases} \tag{4.4.1}$$

L'objectif est de minimiser la puissance totale allouée aux capteurs (ensemble des gains (G_1, ..., G_ℓ, ..., G_L)) tout en gardant la probabilité d'erreur de fusion au dessous d'un seuil requis ϵ.

Deux situations sont à envisager : (1) les observations locales sont i.i.d. ; (2) les observations au niveau des différents nœuds sont corréléess. Dans la situation (1), la formulation du problème postule l'absence de corrélation, et il faut donc vérifier que la matrice de covariance du bruit d'observations est diagonale. Dans ce cas, la solution optimale pour l'allocation des gains est dérivée analytiquement. Dans le situation (2), la matrice de covariance est non diagonale. Une approximation sera donc nécessaire pour se ramener à une matrice tridiagonale. Le modèle de corrélation spatiale rend par conséquent difficile l'obtention d'une expression analytique du problème d'optimisation, dont la résolution s'obtient par une approche numérique.

4.4.1 Observations indépendantes

Si l'on se place dans le cas où les vecteurs des observations recueillies par les capteurs ainsi que les vecteurs de bruits du récepteur sont indépendants et identiquement distribués (i.i.d.), $\mathbf{\Sigma}_v = \sigma_v^2\mathbf{I}$ et $\mathbf{\Sigma}_w = \sigma_w^2\mathbf{I}$, où $\mathbf{I}$ est la matrice identité de dimension $L \times L$.

L'erreur quadratique moyenne (MSE : *mean square error*) basée sur le signal reçu (4.3.3) est donnée par :

$$MSE = \left(\sum_{\ell=1}^{L} \frac{H_\ell^2 G_\ell^2}{\sigma_v^2 H_\ell^2 G_\ell^2 + \sigma_w^2} \right)^{-1} \tag{4.4.2}$$

Par ailleurs, l'expression de la probabilité d'erreur de fusion (4.3.10) peut être simplifiée de la manière suivante :

$$\mathcal{P}_E = \mathcal{Q}\left(\frac{m}{2} \sqrt{\sum_{\ell=1}^{L} \frac{H_\ell^2 G_\ell^2}{\sigma_v^2 H_\ell^2 G_\ell^2 + \sigma_w^2}} \right) \tag{4.4.3}$$

Il est intéressant de noter que la probabilité d'erreur au centre de fusion a un seuil de performance égal à $\mathcal{Q}\left(\frac{\sqrt{L\gamma_0}}{2}\right)$ quand G_ℓ^2 tend vers l'infini ($\ell = 1,2,\ldots,L$), i.e.,

$$\begin{gathered} lim_{G_\ell \to \infty} \sum_{\ell=1}^{L} \frac{H_\ell^2 G_\ell^2}{\sigma_v^2 H_\ell^2 G_\ell^2 + \sigma_w^2} = \frac{L}{\sigma_v^2} \\ lim_{G_\ell \to \infty} \mathcal{P}_E = \mathcal{Q}\left(\frac{\sqrt{L\gamma_0}}{2}\right) \end{gathered} \tag{4.4.4}$$

Intuitivement, l'équation (4.4.4) signifie que, pour un nombre fixé L, la performance obtenue est déterminée principalement par la qualité de l'observation au niveau de chaque nœud, indépendamment de la qualité du canal de transmission sans fil.

En considérant la probabilité d'erreur de fusion (4.4.3), quand les observations locales sont i.i.d., la première inégalité dans (4.4.1) peut être formulée comme suit :

$$\beta \leq \sqrt{\sum_{\ell=1}^{L} \frac{H_\ell^2 G_\ell^2}{\sigma_v^2 H_\ell^2 G_\ell^2 + \sigma_w^2}} \tag{4.4.5}$$

où

$$\beta = \frac{2}{m} Q^{-1}(\epsilon) \tag{4.4.6}$$

Le problème d'optimisation peut alors s'écrire de la façon suivante :

$$\begin{cases} \min \sum_{\ell=1}^{L} G_\ell^2 & \\ \text{s.c.} & \beta^2 - \sum_{\ell=1}^{L} \frac{H_\ell^2 G_\ell^2}{\sigma_v^2 H_\ell^2 G_\ell^2 + \sigma_w^2} \leq 0 \\ & G_\ell \geq 0; \quad \ell = 1, 2, \ldots, L \end{cases} \tag{4.4.7}$$

On note comme suit l'écriture du Lagrangien associé au problème (4.4.7) :

$$\mathfrak{L}(G, \lambda_0, \mu_k) = \sum_{k=1}^{L} G_k^2 + \lambda_0 \left[\beta^2 - \sum_{k=1}^{L} \frac{H_\ell^2 G_\ell^2}{\sigma_v^2 H_\ell^2 G_\ell^{*2} + \sigma_w^2} \right] + \sum_{k=1}^{L} \mu_k (-G_k) \tag{4.4.8}$$

avec $\lambda_0 \geq 0$ et $\mu_k \geq 0$ pour $k = 0, \ldots, L$ sont les multiplicateurs de Lagrange associés aux contraintes d'inégalité du programme d'optimisation. Il y a autant de multiplicateurs qu'il y a de contraintes. Remarquons alors que si la fonction objectif et les contraintes sont convexes, les conditions Karush Kuhn Tucker (KKT) [Kuhn, 1982] sont valides. La solution optimale peut être exprimée comme suit :

$$G_k^2 = \begin{cases} \frac{\sigma_w^2}{H_k^2 \sigma_v^2} \left[\frac{H_k \sum_{j=1}^{K_1} \frac{1}{H_j}}{(K_1 - \beta^2 \sigma_v^2)} - 1 \right] & \text{si } k \leq K_1 \text{ et } L > \beta^2 \sigma_v^2 \\ 0 & \text{if } k > K_1 \text{ et } L > \beta^2 \sigma_v^2 \\ \textit{non réalisable} & \text{si } L < \beta^2 \sigma_v^2 \end{cases} \tag{4.4.9}$$

où le terme K_1 est tel que $f(K_1) < 1$ et $f(K_1 + 1) \geq 1$ pour $1 \leq K_1 \leq L$, $f(k) = \frac{(k - \beta^2 \sigma_v^2)}{H_k \sum_{j=1}^{k} \frac{1}{H_j}}$. La preuve de l'unicité de K_1 et de l'optimalité globale de la solution (4.4.9) du problème d'optimisation (4.4.7) est montrée dans [Wimalajeewa & Jayaweera, 2008].

La solution présentée dans (4.4.9) est réalisable seulement si $L > \beta^2 \sigma_v^2$, i.e., $\gamma_0 > \frac{4}{L} \left(\mathcal{Q}^{-1}(\mathcal{P}_E) \right)^2$, ce qui implique que la probabilité d'erreur $\mathcal{P}_E$ possède une borne inférieure égale à $\mathcal{Q}\left(\frac{\sqrt{L\gamma_0}}{2} \right)$, ce qui est cohérent avec (4.4.4).

Prenons à présent l'hypothèse d'une transmission idéalisée, c'est-à-dire, que les informations envoyées par les capteurs sont supposées être reçues au centre de fusion intactes sans distorsions. Cette hypothèse peut être raisonnable pour certaines applications, mais la réalité est bien

différente. Le signal transmis doit faire face aux pertes de propagation dues à la distance, aux atténuations induites par les obstacles qu'il trouve sur son parcours et aux évanouissements (*fading*) suscités par l'existence de trajets multiples. Contrairement au canal idéalisé, un canal réel est affecté par des altérations et du bruit qui perturbent le signal d'intérêt. Mais l'acquisition des informations sur l'état du canal peut être trop coûteux pour un réseau de capteurs à ressources limitées. Il peut également être impossible d'estimer avec précision le canal qui peut subir des variations principalement dues aux changements des conditions de propagation entre l'émetteur et le récepteur. Par conséquent, il est plus raisonnable de supposer que le centre de fusion soit muni d'une certaine quantité d'informations sur l'état du canal de propagation.

Supposons donc que le centre de fusion a la connaissance préalable des cœfficients de *fading* du canal. En considérant un canal de *Rayleigh*, le *fading* est décrit par une loi complexe gaussienne. L'enveloppe du canal présente alors une distribution de *Rayleigh* de paramètre unité, donnée par :

$$f(x) = \begin{cases} \frac{x}{\sigma^2} exp\left(\frac{-x^2}{2\sigma^2}\right) & \text{si } x \geq 0 \\ 0 & \text{sinon} \end{cases} \tag{4.4.10}$$

Les cœfficients de *fading* H_ℓ ($\ell = 1, \ldots, L$) sont des variables aléatoires gaussiennes complexes, et sans perte de généralité, ces cœfficients sont supposés être classés dans l'ordre décroissant, i.e., $H_1 \geq H_2 \ldots \geq H_L$.

4.4.2 Observations corrélées

Supposer que toutes les observations sont indépendantes est relativement commode pour l'analyse. Toutefois, cette situation, n'est pas la plus courante dans la pratique. Nous nous intéressons donc au cas plus général et pratique où les observations locales sont spatialement corrélées.

On suppose un réseau rectiligne de capteurs équidistants. Nous notons par d la distance euclidienne entre capteurs. La corrélation entre les observations des capteurs i et j est proportionnelle à $\rho^{d|i-j|}$, où $0 < |\rho| \leq 1$. Un cœfficient de corrélation $\rho = 1$ signifie que deux observations sont parfaitement corrélées. Un cœfficient de corrélation $0 < \rho < 1$ indique que deux observations sont partiellement corrélées (i.e., corrélation spatiale), tandis que $\rho = 0$ implique que deux observations sont indépendantes.

La matrice de covariance $\mathbf{\Sigma}_v$ associée au bruit d'observation présente une structuration particulière de type matrice de *Toeplitz*[4] symétrique, aussi appelée matrice de *Kac-Murdock-*

4. Une matrice de *Toeplitz* est définie comme une matrice à diagonales constantes, c'est-à-dire une matrice pour laquelle les éléments sur chaque diagonale descendante de gauche à droite sont égaux.

Szegö [Dow, 2003] :

$$\mathbf{\Sigma}_v = \sigma_v^2 \begin{pmatrix} 1 & \rho^d & \dots & \rho^{d(L-2)} & \rho^{d(L-1)} \\ \rho^d & 1 & \dots & \rho^{d(L-3)} & \rho^{d(L-2)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \rho^{d(L-1)} & \rho^{d(L-2)} & \dots & \rho^d & 1 \end{pmatrix} \tag{4.4.11}$$

Un cas particulier intervient lorsque les observations sont non corréléess : ceci entraîne l'annulation des cœfficients de corrélation et la matrice de covariance se réduit alors à une matrice diagonale d'éléments. Dans le cas corrélé, la matrice de covariance $\mathbf{\Sigma_v}$ n'est pas diagonale, ce qui rend difficile une évaluation analytique de $\mathbf{\Sigma_n^{-1}}$ dans (4.3.10). Ce problème est abordé en faisant une approximation de la matrice $\mathbf{\Sigma}_v$ par une matrice tridiagonale, pour un cœfficient de corrélation ρ suffisamment petit, ce qui correspond au cas où seules les observations de nœuds adjacents sont corrélées. Une telle opération nous permet d'obtenir une expression analytique approchée du problème, et d'en déduire les seuils optimaux.

En suivant la même démarche développée dans [Wimalajeewa & Jayaweera, 2008], nous présentons la borne supérieure de la probabilité d'erreur au centre de fusion en utilisant l'approximation par matrice tridiagonale ainsi que l'inégalité de Bergstrom [Abadir & Magnus, 2005] pour des cœfficients de corrélation suffisamment petits. L'inégalité de Bergstrom est telle que, pour deux matrices symétriques définies positives $\mathbf{P}$ et $\mathbf{Q}$, nous avons :

$$\mathbf{e}^T\mathbf{P}^{-1}\mathbf{e} \geq \frac{\left(\mathbf{e}^T\left(\mathbf{P}+\mathbf{Q}\right)^{-1}\mathbf{e}\right)\left(\mathbf{e}^T\mathbf{Q}^{-1}\mathbf{e}\right)}{\mathbf{e}^T\mathbf{Q}^{-1}\mathbf{e} - \mathbf{e}^T\left(\mathbf{P}+\mathbf{Q}\right)^{-1}\mathbf{e}} \tag{4.4.12}$$

Les équations (4.3.5) et (4.3.10) amènent à, $m^2\mathbf{e}^T\mathbf{A}\mathbf{\Sigma}_n^{-1}\mathbf{A}\mathbf{e} = m^2\mathbf{e}^T\left(\Sigma_v + \sigma_w^2\mathbf{A}^{-2}\right)^{-1}\mathbf{e}$. En prenant $\mathbf{P} = (\Sigma_v + \sigma_w^2\mathbf{A}^{-2})$, la matrice $\mathbf{Q}$ est définie par :

$$\mathbf{Q} = \sigma_v^2 \begin{pmatrix} 1 & -\rho^d & \dots & -\rho^{d(L-2)} & -\rho^{d(L-1)} \\ -\rho^d & 1 & \dots & -\rho^{d(L-3)} & -\rho^{d(L-2)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ -\rho^{d(L-1)} & -\rho^{d(L-2)} & \dots & -\rho^d & 1 \end{pmatrix} \tag{4.4.13}$$

De part l'équation (4.4.12), nous en déduisons que :

$$\mathbf{e}^T\left(\mathbf{\Sigma}_v + \sigma_w^2\mathbf{A}^{-2}\right)^{-1}\mathbf{e} \geq \left(\frac{1}{\sum_{\ell=1}^{L}\frac{H_\ell^2 G_\ell^2}{2\sigma_v^2 H_\ell^2 G_\ell^2 + \sigma_w^2}} - \frac{1}{D}\right)^{-1} \tag{4.4.14}$$

où $D = \mathbf{e}^T\mathbf{Q}^{-1}\mathbf{e}$.

On déduit des équations (4.3.10) et (4.4.14), que la probabilité d'erreur de fusion peut être bornée inférieurement par la quantité :

$$\mathcal{P}_E \leq \mathcal{Q}\left(\frac{m}{2}\left(\frac{1}{\sum_{\ell=1}^{L}\frac{H_\ell^2 G_\ell^2}{2\sigma_v^2 H_\ell^2 G_\ell^2+\sigma_w^2}}-\frac{1}{D}\right)^{-\frac{1}{2}}\right) \tag{4.4.15}$$

Quand $\rho = 0$, $D = L/\sigma_v^2$, nous retrouvons les mêmes résultats que dans le cadre des observations i.i.d., à savoir :

$$\lim_{G_\ell\to\infty}\left(\frac{1}{\sum_{\ell=1}^{L}\frac{H_\ell^2 G_\ell^2}{2\sigma_v^2 H_\ell^2 G_\ell^2+\sigma_w^2}}-\frac{1}{D}\right)=\frac{L}{\sigma_v^2} \tag{4.4.16}$$

Dans le cas des observations corréléess, le problème original d'optimisation (4.4.1) est donc équivalent à :

$$\begin{cases} \min \sum_{\ell=1}^{L} G_\ell^2 & \\ \text{s.c.} & \beta^2 - e^T A \Sigma_n^{-1} A e \leq 0 \\ & G_\ell \geq 0; \quad \ell = 1, 2, \ldots, L \end{cases} \tag{4.4.17}$$

Résoudre ce problème n'est généralement pas possible de manière analytique. Nous avons alors affaire à un problème d'optimisation numérique. Ce problème peut souvent être simplifié en proposant une approximation analytique qui minimise la borne inférieure de la probabilité d'erreur de fusion (4.4.15). Ce qui nous amène au problème d'optimisation suivant :

$$\begin{cases} \min \sum_{\ell=1}^{L} G_\ell^2 & \\ \text{s.c.} & q - \sum_{\ell=1}^{L}\frac{H_\ell^2 G_\ell^2}{2\sigma_v^2 H_\ell^2 G_\ell^2+\sigma_w^2} \leq 0 \\ & G_\ell \geq 0; \quad \ell = 1, 2, \ldots, L \end{cases} \tag{4.4.18}$$

où $q = \left(\frac{1}{\beta^2}+\frac{1}{D}\right)^{-1}$. Ainsi, la puissance optimale allouée aux différents nœuds peut être déterminée de la même manière que dans le cadre des observations indépendantes (cf. section 4.4.1). La solution optimale au problème (4.4.18) est donnée par :

$$G_k^2 = \begin{cases} \frac{\sigma_w^2}{2H_k^2\sigma_v^2}\left[\frac{H_k\sum_{j=1}^{N1}\frac{1}{H_j}}{(N1-2\sigma_v^2 q)}-1\right] & \text{si } k \leq N1 \text{ et } L > 2\sigma_v^2 q \\ 0 & \text{si } k > N1 \text{ et } L > 2\sigma_v^2 q \\ \textit{non réalisable} & \text{si } L < 2\sigma_v^2 q \end{cases} \tag{4.4.19}$$

où $N1$ est unique et défini tel que $\tilde{f}(N1) < 1$ and $\tilde{f}(N1+1) \geq 1$ for $1 \leq N1 \leq L$. $\tilde{f}(k) = \frac{(k-2\sigma_v^2 q)}{H_k\sum_{j=1}^{k}\frac{1}{H_j}}$.

Dans la solution optimale, le nombre de capteurs actifs doit être supérieur à $2\sigma_v^2 q$ pour satisfaire les contraintes liées à la probabilité d'erreur au centre de fusion. Cette solution suggère également que certains capteurs, qui fonctionnent inutilement, se mettent en veille, afin de ne pas gaspiller d'énergie. Ainsi, nous allons pouvoir minimiser le nombre de nœuds actifs et maximiser la durée de vie du réseau.

En d'autres termes, nous visons à calculer le nombre minimal de rapports que doit recevoir le centre de fusion à partir des capteurs pour ne pas dépasser un seuil prédéfini de distorsion maximale tolérée. Idéalement, un protocole de communication pour les RCSF doit faire fonctionner les capteurs uniquement en cas de besoin. Autrement, tous les capteurs doivent être en mode *sommeil*. Nous avons donc besoin de faire fonctionner uniquement les bons capteurs aux bons moments pour transmettre seulement des données utiles.

Nous proposons dans la suite une approche numérique pour la répartition optimale de la puissance lorsque les observations sont arbitrairement corrélées. La solution proposée dans ce travail est basée sur une adaptation de l'algorithme BBO pour les problèmes d'optimisation avec contraintes (cf. chapitre 3).

4.5 Résultats expérimentaux et analyse

4.5.1 Représentation d'une solution

Pour le problème d'allocation optimal de puissance, une solution est représentée par un vecteur de dimension L : $\vec{G} = [G_1, G_2, \ldots, G_L]$ où chaque variable de décision G_ℓ ($\ell = 1, \ldots, L$) désigne le gain de l'amplificateur au niveau du nœud capteur ℓ, directement codé en nombres réels.

4.5.2 Fonction d'évaluation

Dans les sections 4.4.2 et 4.4.1 nous avons présenté les critères permettant d'évaluer la qualité d'une allocation de puissance dans le cas des observations i.i.d. et corrélées, respectivement. Le problème de recherche de l'allocation optimale de puissance qui minimise les gains sous la contrainte d'erreur de fusion minimale est décrit par l'équation (4.4.7) quand les observations sont i.i.d. et par l'équation (4.4.18) quand les observations sont corrélées.

4.5.3 La prise en compte des contraintes

Nous avons présenté dans le chapitre 1 (section 1.8) les principales stratégies de prise en compte des contraintes dans les problèmes d'optimisation. Comme nous l'avons mentionné au chapitre 3 (section 3.2.1), la prise en compte des contraintes adoptée dans le cadre de ce

travail est basée sur le principe de pénalisation. Résoudre le problème d'optimisation avec contraintes consiste alors à le transformer pour se ramener à un problème sans contraintes de la façon suivante : nous modifions la fonction objectif en lui rajoutant un terme positif qui aura pour effet d'en augmenter significativement la valeur (la pénaliser), ceci toutes les fois où les contraintes sont violées. Seules les contraintes d'inégalité sont prises en compte dans le problème d'allocation d'énergie dans les RCSFs.

Le problème de minimisation sans contraintes que nous allons résoudre est donc :

$$F(\vec{G}) = \begin{cases} f(\vec{G}) & \text{si } \psi_j(\vec{G}) \leq 0 \\ f(\vec{G}) + \Phi(\psi^+(\vec{G})) & \text{sinon} \end{cases} \tag{4.5.1}$$

où :

- $f(G) = \sum_{\ell=1}^{L} G_\ell^2$;
- $\psi_1(G) = \beta^2 - \sum_{\ell=1}^{L} \frac{H_\ell^2 G_\ell^2}{\sigma_v^2 H_\ell^2 G_\ell^2 + \sigma_w^2}$ quand les observations sont i.i.d. ;
- $\psi_1(G) = \beta^2 - e^T A \Sigma_n^{-1} A e$ quand les observations sont corrélées ;
- $\psi_{\ell+1}(G) = -G_\ell$ pour $\ell = 1, 2, \ldots, L$; $G = [G_1, G_2, \ldots, G_L]^T$.
- $\Phi(\psi^+(\vec{G})) = \sum_{j=1}^{p} \left(\psi_j^+(\vec{G})\right)^2$ est la fonction de pénalisation ($i = 1, \ldots, p$) et $p = L + 1$ est le nombre des contraintes d'inégalité ;
- $\psi^+(\vec{G}) = \left[\psi_1^+(\vec{G}),\ \psi_2^+(\vec{G}),\ \ldots,\ \psi_p^+(\vec{G})\right]$ est le degré de violation des contraintes (i.e. $\psi_i^+(\vec{G}) = \max\left\{0, \psi_j(\vec{G})\right\}$.

4.5.4 Paramétrage

Trois variantes de l'algorithme BBO ont été testées dans le cadre de cette étude, pour trouver numériquement le schéma optimal d'allocation de puissance dans les RCSFs. Ces variantes, à savoir CBBO, CBBO-DE et 2-Stage-CBBO-DE appliquent des stratégies différentes pour la mise à jour de la population, en plus de certaines adaptations qui les rendent applicables au problème étudié. (cf. chapitre 3, section 3.2).

Dans l'algorithme CBBO [Boussaïd et al., 2012], la population est mise à jour en appliquant successivement la procédure de migration, suivie de la procédure de mutation de façon itérative, comme le préconise l'algorithme BBO de base (voir l'algorithme 1.1). La procédure de mise à jour de la population de l'algorithme CBBO-DE est, elle, décrite dans [Boussaïd et al., 2012]. L'opérateur de mutation, emprunté à la méthode DE [Storn & Price, 1997; Price et al., 2005] remplace celui de la version de base de BBO. La stratégie de mise à jour de la population de l'algorithme 2-Stage-CBBO-DE est similaire à celle décrite dans [Boussaïd et al., 2011a]. La population est mise à jour en appliquant, alternativement d'une itération de l'algorithme à l'autre, BBO et DE (cf. chapitre 2, section 2.4).

Afin d'évaluer les résultats obtenus par les algorithmes que nous proposons, nous nous comparerons sur quelques jeux de données à d'autres algorithmes classiques de la littérature. Nous décrirons brièvement, dans un premier temps, les algorithmes qui nous servirons pour les comparaisons des performances puis nous présenterons les résultats obtenus.

Nous avons comparé les résultats de nos méthodes avec les algorithmes CDE, CGA et CPSO. Ces algorithmes sont des adaptations, au problème d'optimisation avec contraintes, des algorithmes DE, GA et PSO respectivement, développées pour les besoins de notre étude. La prise en compte des contraintes est la même pour tous les algorithmes (cf. section 4.5.3).

La méthode CDE, procède exactement comme l'algorithme original présenté dans [Price et al., 2005], sauf pour l'opérateur de sélection qui est remplacé par la sélection par tournoi binaire décrite dans la section 3.2.4. L'algorithme DE a été expliqué plus en détail dans la section 1.6.2.2. Le schéma de mutation utilisé ici est *DE/rand/1/bin*, car il donne de meilleures performances. Les paramètres F et CR ont été fixés respectivement à 0,5 et 0,9 [Storn & Price, 1997]. Pour les expériences, une population de 100 individus a été utilisée.

Une adaptation de la version standard de PSO (SPSO 07 [5]), appelée CPSO, pour le problème d'optimisation sous contraintes est également mis en œuvre dans le cadre de cette étude. Dans la phase d'initialisation, les positions et les vitesses de tous les individus sont générées aléatoirement. A chaque itération, une particule i ajuste sa position $\vec{G}_i$ et sa vitesse $\vec{V}_i$ pour chaque dimension ℓ de l'espace de recherche, sur la base de la meilleure position qu'elle a trouvé au cours de son vol (aussi appelée *personal best « pbest »*) et la meilleure position atteinte par les particules « voisines » (*global best « gbest »*). Le lecteur peut se référer, pour plus de détail sur la méthode PSO et sur les formules de mise à jour de la position et de la vitesse, à la section 1.6.3.2. Supposant que $\vec{P}_{i,g}$ représente *pbest* de la $i^{\text{ème}}$ particule à la génération g et $\vec{G}_{i,g+1}$ représente la nouvelle position de la particule i à la génération $(g+1)$. Dans la version standard de PSO, $\vec{P}_{i,g+1} = \vec{G}_{i,g+1}$ seulement si $f(\vec{G}_{i,g+1}) < f(\vec{P}_{i,g})$. Tandis que dans CPSO, deux solutions sont comparées sur la base de la règle de faisabilité décrite dans [Deb, 2000]. Autrement dit, $\vec{P}_{i,g}$ sera remplacé par $\vec{G}_{i,g+1}$, sous l'une des conditions suivantes :

1. $\vec{P}_{i,g}$ est non réalisable, mais $\vec{G}_{i,g+1}$ est réalisable,
2. $\vec{P}_{i,g}$ et $\vec{G}_{i,g+1}$ sont tous les deux réalisables, mais $f(\vec{G}_{i,g+1}) < f(\vec{P}_{i,g})$,
3. $\vec{P}_{i,g}$ et $\vec{G}_{i,g+1}$ sont tous les deux non réalisables, mais $Viol(\vec{G}_{i,g+1}) < Viol(\vec{P}_{i,g})$ où $Viol(.)$ est la valeur de violation des contraintes par une solution non réalisable, définie dans l'équation (4.5.2).

5. http ://www.particleswarm.info/Programs.html

$$Viol(\vec{X}_{i,g}) = \sum_{j=1}^{p} \frac{\psi_j^+(\vec{X}_{i,g})}{\psi_{\max}^+(j)} \tag{4.5.2}$$

où $\psi_j^+(\vec{X}_{i,g})$ est le degré de violation de la $j^{\text{ème}}$ contrainte par le $i^{\text{ème}}$ individu $\vec{X}_{i,g}$ (cf. équation 4.5.1) et $\psi_{\max}^+(j) = \max[\psi_j^+(\vec{X}_{i,g})]$ ($i = 1, \ldots, NP$ avec NP la taille de la population) est la valeur maximale du degré de violation de la $j^{\text{ème}}$ contrainte.

La meilleure solution globale *gbest* est mise à jour de façon similaire. Les paramètres de CPSO sont les mêmes que ceux de la version standard (SPSO 07). La taille de la population est fixée à $(10 + 2 * L^2)$, où L est la dimension du problème (nombre de capteurs).

L'algorithme génétique adapté au problème étudié, appelé CGA (pour Constrained Genetic Algorithm), commence par une génération d'une population initiale d'individus, parmi lesquels sont sélectionnés les individus qui participeront aux opérations de croisement et de mutations. Les individus issus de la phase de recombinaison seront insérés, suivant une stratégie de remplacement, dans la nouvelle population. La stratégie de remplacement utilisée prend en considération à la fois la *fitness* et le degrés de violation des contraintes, telle que décrite dans la section 3.2.4. La taille de la population de CGA est de 100. Un croisement binaire simulé (*Simulated Binary Crossover Operator (SBX)*) [Deb & Agrawal, 1995] est appliqué avec un taux de 0,9 et le taux de mutation est fixé à 0,05.

Les algorithmes servant pour la comparaison ont été implémentés en Java au même titre que les différentes versions de BBO.

Les simulations ont été réalisées pour différentes valeurs des différents paramètres du problème, à savoir la probabilité d'erreur de fusion (ϵ), le degré de corrélation (ρ) et le nombre de capteurs (L). Le seuil d'erreur de fusion ϵ prend ses valeurs dans l'ensemble {0,1, 0,05, 0,01, 0,001}, $\rho = \{0,\ 0{,}01,\ 0{,}1,\ 0{,}5\}$ où $\rho = 0$ représente le cas non corrélé. Le rapport signal à bruit (SNR) γ_0 est fixé à 10 dB. Les cœfficients de *fading* H_i du canal suivent une distribution de Rayleigh de paramètre unité (cf. équation 4.4.10). Sans perte de généralité, nous considèrons ces cœfficients classés dans l'ordre décroissant de sorte que $H_1 \geq H_2 \ldots \geq H_L$.

4.5.5 Résultats numériques

Pour effectuer les tests de comparaison sur les différents algorithmes définis précédemment, 30 exécutions de chacun de ces algorithmes ont été effectuées, pour une nombre maximum d'évaluations fixé à 25 000. Nous avons réalisé, pour chaque jeu de données, des statistiques descriptives des résultats trouvés. Ainsi, nous indiquons pour chaque jeu de données la moyenne sur les exécutions du meilleur individu (*mean*), le meilleur sur les 30 exécutions (*best*), et l'écart type (*std.*) entre les différentes exécutions.

Le tableau 4.1, donne une comparaison des résultats obtenus par les algorithmes en compétition, pour différentes valeurs de ϵ et L, dans le cas non corrélé ($\rho = 0$). En terme de meilleures valeurs moyennes, nous pouvons observer que CBBO-DE est classé premier puisqu'il obtient les meilleurs résultats pour $L = 10$ et $L = 50$ capteurs. Pour $L = 20$ capteurs, le 2-Stage-CBBO-DE est meilleur en moyenne sur toutes les instances utilisées. En terme de meilleurs résultats trouvés, nous pouvons observer que CBBO-DE est meilleur que les cinq autres algorithmes en compétition, pour un grand nombre de capteurs, pour les différentes valeurs de probabilité d'erreur au centre de fusion. Pour $L = 20$, CPSO trouve le meilleur résultat pour deux instances et CGA pour une instance sur les quatre instances présentées.

Le tableau 4.2, montre les résultats numériques obtenus par les différents algorithmes lorsque les observations sont corréléess, dans le cas où $L = 10$ capteurs, pour différentes valeurs de la probabilité d'erreur de fusion ϵ et du degré de corrélation ρ. Comme l'illustre ce tableau, CBBO-DE est meilleur en moyenne sur sept instances sur un total de douze. Ce tableau montre également que CGA obtient les meilleurs résultats pour cinq instances, alors que CBBO-DE, 2-Stage-CBBO-DE et CPSO obtiennent les meilleurs résultats pour deux instances et CBBO-DE est meilleur sur trois instances. Dans le cas de faible probabilité d'erreur au centre de fusion, ($\epsilon = 0{,}001$), CGA est la meilleure méthode puisqu'elle obtient les meilleurs résultats pour les différentes valeurs de ρ.

Pour $L = 20$ capteurs, le tableau 4.3 montre que CBBO-DE est la meilleure méthode puisqu'elle obtient les meilleurs résultats en moyenne pour toutes les instances.

Les résultats pour $L = 50$ capteurs, dans le cas d'observations corrélées, sont présentés dans le tableau 4.4. CBBO est la meilleure méthode puisqu'elle obtient les meilleurs résultats en moyenne pour dix instances sur un total de douze. A partir de ces résultats, nous pouvons globalement conclure que, lorsque les observations sont corrélées (tableaux 4.2, 4.3 et 4.4) les performances de l'algorithme CBBO-DE, par rapport aux autres algorithmes, étaient meilleures pour $L = 10$ et $L = 20$ capteurs que pour $L = 50$ capteurs, où la qualité de la solution obtenue par CBBO est supérieure.

Le tableau 4.5 montre le gain de l'amplificateur attribué à chaque capteur pour $L = 10$ capteurs. La deuxième colonne du tableau montre les résultats analytiques de l'allocation optimale de puissance quand les observations sont i.i.d. ($\rho = 0$), où 0 signifie qu'un nœud est dans un mode de sommeil (ou silencieux) n'impliquant que de faibles dépenses d'énergie. Ces résultats analytiques sont obtenus en utilisant la méthode des multiplicateurs de Lagrange [Wimalajeewa & Jayaweera, 2008]. Par la connaissance de l'état du canal, comme cela a été supposé dans la section (4.4.1), le transmetteur peut répartir les puissances entre les capteurs de manière telle à atteindre le niveau de fiabilité voulu. Rappelons que les cœfficients de *fading* H_ℓ ($\ell = 1, \ldots, L$)

du canal Rayleigh sont supposés être classés dans l'ordre décroissant, i.e., $H_1 \geq H_2 \ldots \geq H_L$. Il est alors possible d'exploiter pleinement cette information afin d'optimiser globalement la transmission en anticipant l'effet du canal. La stratégie d'allocation de puissance consiste, comme nous pouvons l'observer dans le tableau 4.5, à distribuer plus de puissance sur les canaux avec de bons cœfficients de *fading*. En revanche, une moindre puissance ou aucune est répartie sur les canaux à faible cœfficient. Par conséquent, nous pouvons décider sur le nombre de capteurs à activer simultanément et ceux qu'on gardera en mode silencieux.

4.6 Conclusion

Dans ce chapitre, nous avons abordé le problème de l'allocation optimale de puissance pour la détection décentralisée d'un signal déterministe dans un réseau de capteurs sans fil. Un schéma optimal d'allocation de puissance ne doit faire intervenir que les capteurs apportant une information significative au processus de décision. En effet, un système utilisant tous les capteurs s'avère très vite irréalisable. Le principe étant de permettre à un certain nombre de nœuds capteurs de se mettre en veille au lieu de rester en mode d'écoute au canal, ce dernier consommant beaucoup d'énergie. L'objectif est donc d'optimiser la consommation énergétique des nœuds afin d'améliorer les performances du réseau et maximiser sa durée de vie. De façon générale, économiser l'énergie revient finalement à trouver le meilleur compromis entre les différentes activités consommatrices en énergie et plus particulièrement, à limiter les phases de communication qui sont les plus coûteuses et qu'il convient d'optimiser.

Pour résoudre le problème d'allocation de puissance, nous avons utilisé la méthode des multiplicateurs de Lagrange, dans le cas d'une suite d'observations indépendantes et identiquement distribuées. Dans le cas d'observations spatialement corrélées, ce problème ne possède pas de solution analytique, d'où l'intérêt d'utiliser un algorithme d'optimisation stochastique, tel que BBO. Dans le cadre de cette étude, nous avons proposé trois versions de l'algorithme BBO, adaptées au problème d'optimisation avec contraintes. Nous nous sommes comparés à des méthodes classiques de la littérature : PSO, DE et GA que nous avons adapté pour la prise en compte des contraintes, pour les besoins de notre étude. La méthode CBBO-DE s'est avérée être meilleure que les autres algorithmes concurrents pour de nombreux cas de test. Il a également été observé que, dans le cas d'un grand nombre de capteurs, CBBO donne de meilleurs résultats.

ϵ		CBBO	CBBO-DE	2-Stage-CBBO-DE	CDE	CGA	CPSO
$L = 10$							
	Mean	3,17935E+00	**3,17263E+00**	3,17271E+00	3,17320E+00	3,21051E+00	3,24232E+00
0,1	Std,	4,57E-03	2,00E-04	2,65E-04	6,51E-04	2,84E-02	1,14E-01
	Best	3,17249E+00	3,17233E+00	3,17239E+00	3,17241E+00	3,17971E+00	**3,17230E+00**
	Mean	5,98759E+00	**5,97219E+00**	5,97222E+00	5,97237E+00	5,99619E+00	6,04440E+00
0,05	Std,	9,55E-03	1,21E-05	3,35E-05	1,06E-04	5,88E-02	1,57E-01
	Best	5,97339E+00	5,97218E+00	5,97218E+00	5,97221E+00	**5,69023E+00**	5,97221E+00
	Mean	1,51470E+01	**1,51303E+01**	1,51303E+01	1,51304E+01	1,51400E+01	1,53088E+01
0,01	Std,	7,95E-03	3,97E-05	2,93E-05	8,73E-05	1,14E-01	2,33E-01
	Best	1,51315E+01	**1,51303E+01**	1,51303E+01	1,51303E+01	1,45632E+01	1,51303E+01
	Mean	4,00000E+01	4,00000E+01	4,00000E+01	4,00000E+01	**3,99942E+01**	4,00000E+01
0,001	Std,	0,00E+00	0,00E+00	0,00E+00	0,00E+00	3,09E-02	0,00E+00
	Best	4,00000E+01	4,00000E+01	4,00000E+01	4,00000E+01	**3,98277E+01**	4,00000E+01
$L = 20$							
	Mean	1,94366E+00	1,93989E+00	**1,93780E+00**	1,93974E+00	1,96652E+00	2,30705E+00
0,1	Std,	5,71E-03	8,72E-03	3,51E-03	3,19E-03	1,65E-02	6,06E-01
	Best	1,93454E+00	1,93265E+00	1,93325E+00	1,93487E+00	1,94606E+00	**1,93249E+00**
	Mean	3,65694E+00	3,65123E+00	**3,64836E+00**	3,65344E+00	3,68518E+00	3,88634E+00
0,05	Std,	1,03E-02	9,77E-03	3,00E-03	5,18E-03	1,16E-02	2,44E-01
	Best	3,64379E+00	**3,64179E+00**	3,64406E+00	3,64553E+00	3,66486E+00	3,64469E+00
	Mean	9,12602E+00	9,12339E+00	**9,11030E+00**	9,12452E+00	9,18365E+00	9,24245E+00
0,01	Std,	1,35E-02	1,55E-02	6,17E-03	1,01E-02	3,74E-02	2,11E-01
	Best	9,10347E+00	9,10605E+00	9,10173E+00	9,10742E+00	**9,09650E+00**	9,09706E+00
	Mean	2,16507E+01	2,16480E+01	**2,16406E+01**	2,16622E+01	2,17666E+01	2,25538E+01
0,001	Std,	1,87E-02	2,95E-02	1,37E-02	2,11E-02	4,37E-02	1,27E+00
	Best	2,16205E+01	2,16116E+01	2,16250E+01	2,16340E+01	2,17037E+01	**2,15973E+01**
$L = 50$							
	Mean	9,05946E-01	**8,73121E-01**	1,00626E+00	1,05196E+00	9,40606E-01	1,67135E+00
0,1	Std,	1,30E-02	8,70E-03	3,97E-02	5,69E-02	1,83E-02	1,40E+00
	Best	8,84219E-01	**8,67229E-01**	9,38443E-01	9,68506E-01	9,15533E-01	8,80470E-01
	Mean	1,71838E+00	**1,67661E+00**	1,84009E+00	1,91955E+00	1,77156E+00	3,12401E+00
0,05	Std,	1,38E-02	6,22E-03	3,58E-02	7,73E-02	2,51E-02	1,65E+00
	Best	1,69353E+00	**1,66688E+00**	1,76722E+00	1,79779E+00	1,72290E+00	1,85115E+00
	Mean	4,41536E+00	**4,38484E+00**	4,63694E+00	4,74516E+00	4,53181E+00	5,69718E+00
0,01	Std,	2,80E-02	4,76E-02	6,48E-02	9,81E-02	4,15E-02	1,05E+00
	Best	4,37703E+00	**4,34752E+00**	4,53116E+00	4,58753E+00	4,46010E+00	4,46465E+00
	Mean	1,00421E+01	**1,00120E+01**	1,04841E+01	1,06984E+01	1,03002E+01	1,11533E+01
0,001	Std,	3,72E-02	4,17E-02	1,04E-01	1,11E-01	6,43E-02	1,25E+00
	Best	9,96900E+00	**9,91934E+00**	1,02832E+01	1,05173E+01	1,02028E+01	9,97487E+00

TABLEAU 4.1: Résultats numériques quand les observations sont i.i.d. ($\rho = 0$), $L = \{10, 20, 50\}$ capteurs, $\gamma_0 = 10$ dB et $\epsilon = \{0{,}1,\ 0{,}05,\ 0{,}01,\ 0{,}001\}$.

ϵ		CBBO	CBBO-DE	2-Stage-CBBO-DE	CDE	CGA	CPSO
$\rho = 0{,}01$							
	Mean	3,19130E+00	**3,18336E+00**	3,18356E+00	3,18470E+00	3,21675E+00	3,26267E+00
0,1	Std,	5,02E-03	5,02E-03	3,94E-04	6,74E-04	1,72E-02	1,28E-01
	Best	3,18434E+00	3,18305E+00	3,18307E+00	3,18322E+00	3,18874E+00	**3,18300E+00**
	Mean	6,01119E+00	**5,99738E+00**	5,99740E+00	5,99758E+00	6,00321E+00	6,10566E+00
0,05	Std,	1,06E-02	1,89E-05	4,78E-05	1,40E-04	1,18E-01	2,23E-01
	Best	5,99912E+00	5,99736E+00	5,99758E+00	5,99742E+00	**5,37139E+00**	5,99748E+00
	Mean	1,52741E+01	**1,52553E+01**	1,52553E+01	1,52554E+01	1,53046E+01	1,54702E+01
0,01	Std,	8,98E-03	9,41E-05	4,00E-05	7,17E-05	2,89E-02	3,62E-01
	Best	1,52588E+01	**1,52553E+01**	1,52553E+01	1,52553E+01	1,52680E+01	1,52553E+01
	Mean	4,00000E+01	4,00000E+01	4,00000E+01	4,00000E+01	**3,99859E+01**	4,00000E+01
0,001	Std,	0,00E+00	0,00E+00	0,00E+00	0,00E+00	7,00E-02	0,00E+00
	Best	4,00000E+01	4,00000E+01	4,00000E+01	4,00000E+01	**3,96102E+01**	4,00000E+01
$\rho = 0{,}1$							
	Mean	3,29369E+00	**3,28340E+00**	3,28385E+00	3,28434E+00	3,31745E+00	3,32138E+00
0,1	Std,	6,64E-03	1,12E-04	2,99E-04	7,08E-04	2,33E-02	8,00E-02
	Best	3,28524E+00	3,28325E+00	3,28329E+00	3,28352E+00	3,29679E+00	**3,28321E+00**
	Mean	6,25118E+00	6,23910E+00	**6,23901E+00**	6,23930E+00	6,27467E+00	6,28582E+00
0,05	Std,	5,71E-03	3,85E-05	1,76E-05	3,13E-04	2,15E-02	5,08E-02
	Best	6,24318E+00	6,23900E+00	**6,23898E+00**	6,23910E+00	6,24694E+00	6,23913E+00
	Mean	1,65806E+01	**1,65620E+01**	1,65624E+01	1,65625E+01	1,65821E+01	1,68063E+01
0,01	Std,	1,03E-02	3,56E-09	1,08E-04	1,99E-04	6,88E-02	4,75E-01
	Best	1,65677E+01	1,65620E+01	1,65623E+01	1,65623E+01	**1,62204E+01**	1,65623E+01
	Mean	4,00000E+01	4,90770E+01	4,00000E+01	4,00000E+01	**3,99607E+01**	4,00000E+01
0,001	Std,	4,77E-07	7,81E-04	0,00E+00	0,00E+00	2,08E-01	0,00E+00
	Best	4,00000E+01	4,90210E+01	4,00000E+01	4,00000E+01	**3,88388E+01**	4,00000E+01
$\rho = 0{,}5$							
	Mean	3,87091E+00	**3,85830E+00**	3,85885E+00	3,85995E+00	3,90135E+00	3,90053E+00
0,1	Std,	6,49E-03	2,32E-04	4,53E-04	9,29E-04	2,64E-02	4,11E-02
	Best	3,86102E+00	**3,85800E+00**	3,85827E+00	3,85840E+00	3,86301E+00	3,85817E+00
	Mean	8,16317E+00	8,13610E+00	**8,13608E+00**	8,13644E+00	8,18067E+00	8,18731E+00
0,05	Std,	1,81E-02	8,04E-05	9,41E-05	3,28E-04	2,16E-02	9,29E-02
	Best	8,13971E+00	8,13600E+00	**8,13596E+00**	8,13604E+00	8,14816E+00	8,13605E+00
	Mean	3,49787E+01	**3,43490E+01**	3,49497E+01	3,49508E+01	3,50209E+01	3,51627E+01
0,01	Std,	1,89E-02	8,08E-07	4,45E-04	9,72E-04	4,67E-02	1,43E-01
	Best	3,49545E+01	**3,43480E+01**	3,49490E+01	3,49492E+01	3,48397E+01	3,49499E+01
	Mean	4,00000E+01	4,00000E+01	4,00000E+01	4,00000E+01	**3,99640E+01**	4,00000E+01
0,001	Std,	0,00E+00	0,00E+00	0,00E+00	8,26E-07	1,32E-01	0,00E+00
	Best	4,00000E+01	4,00000E+01	4,00000E+01	4,00000E+01	**3,92740E+01**	4,00000E+01

TABLEAU 4.2: Comparaison des résultats numériques quand les observations sont corrélées : $\rho = \{0{,}01,\ 0{,}1,\ 0{,}5\}$, $L = 10$ capteurs, $\gamma_0 = 10$ dB et $\epsilon = \{0{,}1,\ 0{,}05,\ 0{,}01,\ 0{,}001\}$,

ϵ		CBBO	CBBO-DE	2-Stage-CBBO-DE	CDE	CGA	CPSO
$\rho = 0{,}01$							
	Mean	1,94926E+00	**1,93960E+00**	1,94247E+00	2,01270E+00	1,96992E+00	2,19232E+00
0,1	Std,	9,26E-03	2,14E-03	2,14E-03	1,51E-02	1,24E-02	2,45E-01
	Best	1,94131E+00	1,93760E+00	1,93938E+00	1,98330E+00	1,95234E+00	**1,93730E+00**
	Mean	3,67103E+00	**3,65590E+00**	3,66186E+00	3,68830E+00	3,70072E+00	3,88189E+00
0,05	Std,	7,16E-03	9,74E-04	3,08E-03	3,05E-02	1,70E-02	2,47E-01
	Best	3,66106E+00	**3,65480E+00**	3,65729E+00	3,65620E+00	3,67640E+00	3,65888E+00
	Mean	9,19023E+00	**9,16070E+00**	9,17459E+00	9,18835E+00	9,22620E+00	9,41124E+00
0,01	Std,	1,18E-02	8,98E-04	4,59E-03	9,41E-03	1,25E-01	4,45E-01
	Best	9,17263E+00	9,15980E+00	9,16595E+00	9,17181E+00	**8,57142E+00**	9,15913E+00
	Mean	2,18956E+01	**2,18420E+01**	2,18784E+01	2,19148E+01	2,20025E+01	2,22904E+01
0,001	Std,	2,17E-02	3,99E-03	9,98E-03	2,30E-02	4,00E-02	8,45E-01
	Best	2,18605E+01	**2,18400E+01**	2,18591E+01	2,18677E+01	2,19378E+01	2,18406E+01
$\rho = 0{,}1$							
	Mean	2,00287E+00	**1,99050E+00**	1,99530E+00	1,99980E+00	2,02247E+00	2,26079E+00
0,1	Std,	1,03E-02	1,32E-03	2,06E-03	5,18E-03	1,52E-02	2,22E-01
	Best	1,99094E+00	**1,98930E+00**	1,99199E+00	1,99325E+00	1,99851E+00	1,99635E+00
	Mean	3,81559E+00	**3,79580E+00**	3,80405E+00	3,80747E+00	3,84429E+00	4,07500E+00
0,05	Std,	8,52E-03	1,89E-03	3,07E-03	6,88E-03	1,62E-02	2,79E-01
	Best	3,80046E+00	3,79400E+00	3,79833E+00	3,79842E+00	3,82339E+00	**3,79367E+00**
	Mean	9,82097E+00	**9,78940E+00**	9,80599E+00	9,81877E+00	9,87166E+00	9,93990E+00
0,01	Std,	1,74E-02	9,19E-04	6,07E-03	9,16E-03	2,32E-02	2,00E-01
	Best	9,79300E+00	9,78840E+00	9,79575E+00	9,80198E+00	9,82768E+00	**9,78741E+00**
	Mean	2,43796E+01	**2,43240E+01**	2,43596E+01	2,43919E+01	2,45121E+01	2,51887E+01
0,001	Std,	2,30E-02	1,96E-03	1,02E-02	1,77E-02	5,04E-02	1,13E+00
	Best	2,43481E+01	**2,43230E+01**	2,43418E+01	2,43649E+01	2,43882E+01	2,43241E+01
$\rho = 0{,}5$							
	Mean	2,31219E+00	**2,30260E+00**	2,31333E+00	2,31651E+00	2,33887E+00	2,49607E+00
0,1	Std,	6,19E-03	2,19E-03	4,87E-03	6,61E-03	1,41E-02	2,50E-01
	Best	2,30151E+00	**2,30070E+00**	2,30616E+00	2,30666E+00	2,31180E+00	2,30100E+00
	Mean	4,86493E+00	**4,83570E+00**	4,86189E+00	4,87636E+00	4,91358E+00	5,13300E+00
0,05	Std,	1,74E-02	2,71E-03	1,27E-02	1,04E-02	1,91E-02	3,81E-01
	Best	4,84294E+00	**4,83300E+00**	4,84199E+00	4,85172E+00	4,87008E+00	4,83300E+00
	Mean	1,59311E+01	**1,58650E+01**	1,59416E+01	1,59902E+01	1,59949E+01	1,60754E+01
0,01	Std,	2,17E-02	3,27E-03	2,47E-02	3,43E-02	1,41E-01	1,82E-01
	Best	1,59008E+01	1,58620E+01	1,59021E+01	1,59305E+01	**1,55192E+01**	1,58611E+01
	Mean	6,16194E+01	**6,06850E+01**	6,16165E+01	6,16119E+01	6,16867E+01	6,59741E+01
0,001	Std,	1,47E-02	6,51E-02	1,77E-02	8,82E-03	1,98E-01	2,93E+00
	Best	6,15896E+01	**6,05660E+01**	6,15852E+01	6,15966E+01	6,06504E+01	6,17442E+01

TABLEAU 4.3: Comparaison des résultats numériques quand les observations sont corrélées : $\rho = \{0{,}01,\ 0{,}1,\ 0{,}5\}$, $L = 20$ capteurs, $\gamma_0 = 10$ dB et $\epsilon = \{0{,}1,\ 0{,}05,\ 0{,}01,\ 0{,}001\}$,

ϵ		CBBO	CBBO-DE	2-Stage-CBBO-DE	CDE	CGA	CPSO
$\rho = 0{,}01$							
	Mean	**9,02118E-01**	1,48094E+00	9,96784E-01	1,04841E+00	9,44415E-01	1,87671E+00
0,1	Std,	7,89E-03	4,68E-01	3,22E-02	4,42E-02	1,88E-02	1,36E+00
	Best	8,88871E-01	8,79589E-01	9,42663E-01	9,62202E-01	9,14747E-01	**8,76651E-01**
	Mean	**1,72662E+00**	2,86366E+00	1,84437E+00	1,92668E+00	1,78391E+00	3,44436E+00
0,05	Std,	1,63E-02	8,37E-01	3,30E-02	7,91E-02	2,36E-02	1,75E+00
	Best	**1,69629E+00**	1,69376E+00	1,77337E+00	1,78682E+00	1,74011E+00	1,68829E+00
	Mean	**4,44819E+00**	6,05320E+00	4,67165E+00	4,79533E+00	4,57978E+00	5,69284E+00
0,01	Std,	2,27E-02	7,98E-01	5,41E-02	1,04E-01	3,94E-02	9,10E-01
	Best	4,40527E+00	**4,39060E+00**	4,55601E+00	4,63914E+00	4,46567E+00	4,39809E+00
	Mean	**1,01404E+01**	1,07460E+01	1,05199E+01	1,07251E+01	1,03588E+01	1,12213E+01
0,001	Std,	3,91E-02	3,31E-01	6,85E-02	1,35E-01	6,17E-02	1,41E+00
	Best	1,00660E+01	1,01680E+01	1,03465E+01	1,04958E+01	1,02185E+01	**1,00242E+01**
$\rho = 0{,}1$							
	Mean	**9,34620E-01**	3,08130E+00	1,03799E+00	1,08992E+00	9,80039E-01	1,74470E+00
0,1	Std,	1,14E-02	4,30E-01	3,06E-02	5,93E-02	1,76E-02	1,08E+00
	Best	**9,15381E-01**	2,43080E+00	1,11310E+00	9,97765E-01	9,35467E-01	9,80038E-01
	Mean	**1,80867E+00**	4,37700E+00	1,94203E+00	2,01247E+00	1,88165E+00	2,88210E+00
0,05	Std,	1,06E-02	6,28E-01	3,55E-02	7,45E-02	2,94E-02	1,15E+00
	Best	**1,78997E+00**	3,37890E+00	1,86992E+00	1,88792E+00	1,83228E+00	1,79232E+00
	Mean	**4,75915E+00**	6,65320E+00	4,98549E+00	5,10913E+00	4,89356E+00	5,96645E+00
0,01	Std,	2,04E-02	1,01E+00	6,12E-02	7,77E-02	4,51E-02	1,17E+00
	Best	**4,71623E+00**	5,49050E+00	4,82053E+00	4,98578E+00	4,80740E+00	4,89258E+00
	Mean	**1,09317E+01**	1,16690E+01	1,14322E+01	1,16480E+01	1,12121E+01	1,16309E+01
0,001	Std,	3,99E-02	5,81E-01	1,15E-01	1,56E-01	7,61E-02	8,29E-01
	Best	1,08731E+01	1,10910E+01	1,12102E+01	1,14085E+01	1,09675E+01	**1,07873E+01**
$\rho = 0{,}5$							
	Mean	1,62230E+00	1,67224E+00	**1,23836E+00**	1,60562E+00	1,55067E+00	2,00899E+00
0,1	Std,	1,80E-01	3,81E-01	3,62E-02	3,35E-01	4,21E-01	1,57E+00
	Best	1,38950E+00	1,11521E+00	1,19164E+00	1,17440E+00	1,10691E+00	**1,10187E+00**
	Mean	2,77030E+00	3,24208E+00	**2,52436E+00**	3,12709E+00	3,03643E+00	3,18564E+00
0,05	Std,	1,51E-01	6,00E-01	3,95E-02	5,28E-01	6,66E-01	1,33E+00
	Best	2,56400E+00	2,36221E+00	2,45759E+00	2,48189E+00	**2,35095E+00**	2,38701E+00
	Mean	**7,19150E+00**	7,42100E+00	7,22594E+00	8,20864E+00	8,03350E+00	7,80258E+00
0,01	Std,	1,50E-01	3,24E-01	6,71E-02	9,17E-01	1,20E+00	1,04E+00
	Best	6,96500E+00	6,91000E+00	7,02768E+00	7,17023E+00	6,78723E+00	**6,77276E+00**
	Mean	**1,88410E+01**	1,84490E+01	1,92334E+01	2,05440E+01	2,02199E+01	1,91415E+01
0,001	Std,	1,20E-01	3,68E-02	1,68E-01	1,12E+00	1,73E+00	4,94E-01
	Best	1,86090E+01	**1,83800E+01**	1,89643E+01	1,92448E+01	1,84280E+01	1,84879E+01

TABLEAU 4.4: Comparaison des résultats numériques quand les observations sont corrélées : $\rho = \{0{,}01, 0{,}1, 0{,}5\}$, $L = 50$ capteurs, $\gamma_0 = 10$ dB et $\epsilon = \{0{,}1,\ 0{,}05,\ 0{,}01,\ 0{,}001\}$.

Sensors	Analytical	CBBO	CBBO-DE	2-Stage-CBBO-DE	CDE	CGA	CPSO
$\rho = 0$							
S1	1,0362	1,0461	1,0379	1,0362	1,0376	0,9988	1,036
S2	0,9972	0,9995	0,9967	0,9992	0,9982	0,9612	0,9976
S3	0,8834	0,8833	0,8842	0,8879	0,8819	0,9542	0,8829
S4	0,4823	0,4784	0,4781	0,4783	0,4819	0,4735	0,4805
S5	0,3021	0,2641	0,3009	0,2866	0,2968	0,3077	0,3053
S6	0	0,0196	0,0252	0,0258	0,0301	0,1352	0,0033
S7	0	0,0044	0,0131	0	0,0085	0,0985	0,0127
S8	0	1,11E-08	0,0035	0,0025	0	0,009	0
S9	0	9,11E-04	3,90E-05	0	0,0096	5,22E-04	4,45E-04
S10	0	0,0035	0,0017	0,0036	0,0019	0,023	3,25E-04
Poptimal	3,1723	3,1725	3,1723	3,1723	3,1724	3,1797	3,1723
$\rho = 0{,}01$							
S1		1,0170	1,0387	1,0355	1,0389	1,0047	1,0381
S2		0,9884	0,9959	0,9916	0,9894	1,0346	0,9905
S3		0,9079	0,8833	0,8908	0,8824	0,8842	0,8843
S4		0,4842	0,4816	0,4877	0,4732	0,5196	0,4842
S5		0,3191	0,3146	0,3093	0,3483	0,1860	0,3283
S6		0,0368	0,0359	0,0082	0,0270	0,0154	0,0026
S7		0,1062	0,0038	0,0169	0,0206	0,1470	0,0058
S8		1,46E-08	0,0043	4,52E-04	0,0088	0,0114	2,04E-04
S9		4,02E-10	0,0030	0	9,91E-04	0,0216	1,73E-04
S10		1,04E-08	0,0018	0,0021	0	0,0041	0,0019
Poptimal		3,1843	3,1830	3,1831	3,1832	3,1887	3,1830
$\rho = 0{,}1$							
S1		1,0470	1,0422	1,0439	1,0460	1,0909	1,0460
S2		0,9783	1,0422	0,9635	0,9566	0,9850	0,9625
S3		0,8558	0,8781	0,8784	0,8774	0,7900	0,8754
S4		0,4833	0,5156	0,5090	0,5207	0,4292	0,5088
S5		0,4087	0,4474	0,4564	0,4583	0,4104	0,4494
S6		0,1606	0,1652	0,1589	0,1207	0,3136	0,1883
S7		0,2704	0,0155	0,0297	0,0933	0,2444	9,12E-05
S8		1,46E-08	1,95E-07	5,75E-04	0	0,0328	0
S9		3,34E-09	2,92E-09	4,69E-04	0	0,0086	3,56E-05
S10		2,83E-09	0	0,0051	0	0,0199	0
Poptimal		3,2852	3,2832	3,2833	3,2835	3,2968	3,2832
$\rho = 0{,}5$							
S1		1,1575	1,1331	1,1346	1,1309	1,0956	1,1332
S2		0,8641	0,8202	0,8218	0,8375	0,7977	0,8334
S3		0,7925	0,8655	0,8579	0,8633	0,8290	0,8456
S4		0,7110	0,1630	0,2273	0,1443	0,0462	0,2466
S5		0,6426	0,6246	0,6178	0,6078	0,8519	0,5773
S6		0,4774	0,4436	0,4932	0,4826	0,2903	0,5087
S7		8,99E-09	0,7289	0,6948	0,7138	0,7070	0,7152
S8		1,53E-09	0	0,0093	0,0057	0,0027	2,32E-05
S9		4,51E-09	4,98E-04	0,0062	0,0064	0,0874	6,46E-04
S10		3,95E-09	0	0,0172	0	0,1389	0,0018
Poptimal		3,8610	3,8580	3,8583	3,8584	3,8630	3,8582

TABLEAU 4.5: Les résultats analytiques et numériques de l'allocation optimale des gains dans le cas d'observations i.i.d. ($\rho = 0$) et spatialement corrélés : $\rho = \{0{,}01, 0{,}1, 0{,}5\}$. $L = 10$ capteurs. $\gamma_0 = 10$ dB et $\epsilon = 0{,}1$.

Chapitre Cinq

Élaboration de *DBBO-Fuzzy* : Application en segmentation d'images

5.1 Introduction

La représentation et le traitement des images numériques font l'objet de recherches très actives à l'heure actuelle. Les traitements sont nombreux, et parmi eux on compte la *segmentation d'image* qui représente l'une des phases les plus importantes dans la chaîne d'analyse d'images. La qualité des résultats obtenus à l'issue de cette opération détermine la qualité globale de tout le système d'interprétation. Si l'on se réfère à la littérature dans ce domaine, on se rend compte que ce problème est difficile et qu'il est loin d'être résolu.

La segmentation d'image est le processus de décomposition d'une image en un ensemble de régions connexes, homogènes et bien séparées. Ces régions possèdent une certaine uniformité pour une ou plusieurs caractéristiques, telles que la luminance (niveau de gris), la couleur ou la texture, et sont différentes pour au moins une de ses caractéristiques des régions voisines [Freixenet et al., 2002]. L'une des finalités de la segmentation d'image est la reconnaissance de formes, d'objets ou de personnes. La segmentation d'image ainsi définie est un domaine vaste où l'on retrouve de très nombreuses approches [Sahoo et al., 1988; Pal & Pal, 1993]. Une approche particulièrement intéressante est la méthode de *seuillage* (*thresholding*). Une étude exhaustive des techniques de seuillage d'images peut être trouvée dans [Sezgin & Sankur, 2004]. On distingue deux types de seuillage : le *seuillage à deux niveaux*, qui répartit les pixels de l'image en deux régions (l'objet et le fond) selon que leur luminance est supérieure ou non à un seuil spécifié T, et le *seuillage multi-niveaux* (*multithresholding*) qui sépare les pixels en plusieurs classes. Par conséquent, plusieurs seuils doivent être déterminés pour segmenter l'image en régions de luminosité pouvant correspondre à un fond et plusieurs objets. Ce chapitre se focalise entièrement sur ce deuxième type de seuillage et a pour vocation de présenter, après avoir posé quelques définitions essentielles, les diverses techniques pouvant être mises en œuvre.

Toutefois, l'information et les connaissances utilisées en traitement et interprétation des images sont le plus souvent imparfaites. Ces imperfections trouvent leur origine à différents niveaux : phénomènes observés, limites des capteurs et acquisitions des données, bruit, processus numériques de reconstruction, nature des images et mode de représentation de leurs éléments

constitutifs, etc. Face aux imperfections relatives aux images, nous pouvons choisir plusieurs attitudes. Une première attitude consiste à éliminer les imperfections autant que possible. Cela passe par exemple par l'amélioration des capteurs et la multiplication des acquisitions. Une autre approche, qui est généralement adoptée pour la segmentation d'image, consiste à simplement tolérer les imperfections, en produisant des algorithmes robustes, et en y associant des procédures destinées à réparer les échecs. Il existe à cet effet différentes théories décrivant la modélisation des connaissances imparfaites. Les principales sont les *probabilités*, la *théorie des fonctions de croyances* (ou théorie de *Dempster-Shafer* [Shafer, 1976]) et les *sous-ensembles flous* associés à la *théorie des possibilités* [Zadeh, 1965].

Dans ce chapitre, nous nous intéressons en particulier à la théorie des sous-ensembles flous. Cette théorie, proposée par Zadeh [Zadeh, 1965], est un outil mathématique pour analyser l'*imprécision* et l'*incertitude* inhérente à la prise de décisions. L'utilisation de la logique floue se répand de plus en plus pour la segmentation d'image. Cela est en particulier dû au fait que la théorie des sous-ensembles flous est bien adaptée pour décrire des objets dont les frontières sont mal définies. De nombreuses techniques de seuillage basées sur la logique floue ont été mises au point ; la plupart d'entre elles nécessitent l'optimisation d'une certaine fonction critère (par exemple la maximisation de l'entropie floue) pour sélectionner un seuil optimal [Tao et al., 2007; Tobias & Seara, 2002; Zhao et al., 2001]. Ainsi l'appartenance des pixels à une classe de l'image n'est plus absolue mais graduelle. Des critères utilisant la théorie des ensembles flous et des mesures de probabilité seront présentés et discutés. Un algorithme de seuillage à trois niveaux d'images monochromes, appelé DBBO-Fuzzy, a été développé [Boussaïd et al., 2013a]. Cet algorithme est basé sur l'hybridation des algorithmes BBO et l'algorithme d'évolution différentielle (DE). L'approche présentée dans [Tao et al., 2003] est adoptée comme support de base pour ce travail. L'algorithme proposé est testé sur un ensemble d'images de référence, et les résultats expérimentaux montrent son efficacité.

5.2 Notions de base

5.2.1 Sous-ensembles flous : définition et éléments caractéristiques

Les informations que nous avons à traiter en permanence sont essentiellement imparfaites. Elles peuvent être incertaines (nous avons un doute sur leur validité) ou imprécise (nous avons du mal à les exprimer clairement). Il convient avant tout de distinguer les emplois des notions d'imprécision et d'incertitude.

L'imprécision d'une information est souvent, abusivement, confondue avec l'incertitude [Bloch et al., 2003]. La notion d'imprécision est rattachée directement au contenu de l'information ;

elle est définie comme un défaut quantitatif de la connaissance, tandis que l'incertitude est relative à la vérité d'une information, caractérisant sa conformité à la réalité et porte donc sur un défaut qualitatif de la connaissance [Dubois & Prade, 1988]. Un exemple d'imprécision, c'est quand quelqu'un nous dit « *je vais rentrer tard ce soir* ». Une autre situation imprécise peut être remarquée dans une simple conversation où une personne dit qu'*elle n'a pas payé trop cher pour un livre*. L'incertitude peut se traduire par le fait que nous ne puissions pas répondre à une question dans un contexte donné. Par exemple, nous ne savons pas si une proposition est vraie ou fausse, ou si un événement va se produire. Voici des phrases qui sont entachées d'incertitude : « *Il est très possible qu'il neige demain* », « *Il n'est pas absolument certain que Jean vienne à la réunion* ». Il peut y avoir plusieurs causes à l'incertitude, La variabilité d'un phénomène empêche de prévoir le résultat de la prochaine expérience, le manque d'information ou la présence d'informations contradictoires. De plus, la principale cause de l'incertitude d'une information provient de l'imprécision de l'information [Smets, 1996]. En effet, dans le cas d'informations quantitatives, l'imprécision d'une donnée entraîne une incertitude sur l'information véhiculée. Par exemple dire qu'il pleut 10 mm^3/h alors qu'il y en a 15, provoque une incertitude sur le temps qu'il fait. De même, l'incertitude peut induire l'imprécision. Par exemple, le doute sur l'arrivée de la lettre demain peut entraîner une imprécision sur l'estimation du nombre de lettres qui arriveront demain. Ces deux imperfections sont en outre souvent présentes simultanément.

La théorie des sous-ensembles flous (*fuzzy sets theory*) [1] permet de caractériser et de quantifier l'incertitude et l'imprécision liées à des connaissances. Cette théorie se veut une extension de la théorie des ensembles classiques (*crisp set*). Elle remonte à 1965, lorsque Lotfi Zadeh, professeur à Berkeley, a publié son article fondateur sur la théorie des ensembles flous et la logique associée, à savoir la logique floue [Zadeh, 1965].

Cette section présente de manière très synthétique les éléments de la théorie des sous-ensembles flous, telle que proposée par Zadeh [Zadeh, 1965], ainsi que les principaux concepts de base associés. Une description plus complète est disponible dans les ouvrages de référence [Kaufmann, 1975; Zimmermann, 1991; Bouchon-Meunier & Marsala, 2003].

5.2.1.1 Théorie classique des ensembles

Les ensembles classiques sont également appelés ensembles nets, par opposition à flou, et de même la logique classique est également appelée logique booléenne ou binaire. La caractéristique fondamentale d'un ensemble classique est la frontière abrupte entre deux catégories d'éléments : ceux qui appartiennent à l'ensemble et ceux qui n'appartiennent pas à cet ensemble ;

1. Les sous-ensembles flous [Kaufmann, 1975] sont souvent appelés plus simplement ensembles flous. Nous parlons ainsi de la théorie des ensembles flous.

ils appartiennent plutôt à son complémentaire.

Soit $\mathcal{X}$, un ensemble appelé univers du discours ou espace de référence, qui regroupe une collection d'éléments notés x_i : $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$. $\mathcal{X}$ est l'ensemble de toutes les valeurs possibles de x_i, l'ensemble des réels $\mathbb{R}$ dans la plupart des cas [2].

Définition 5.2.1. Un *sous-ensemble classique (ou net)* $\mathcal{A}$ de $\mathcal{X}$ est défini par une fonction caractéristique μ qui prend la valeur 0 pour les éléments de $\mathcal{X}$ n'appartenant pas à $\mathcal{A}$ et la valeur 1 dans le cas contraire. Ainsi, un sous-ensemble classique $\mathcal{A}$ de $\mathcal{X}$ peut s'écrire à partir de sa *fonction caractéristique* $\mu_{\mathcal{A}}$ par :

$$\mu_{\mathcal{A}}(x_i) = \begin{cases} 1 & si\, x_i \in \mathcal{A} \\ 0 & si\, x_i \notin \mathcal{A} \end{cases} \tag{5.2.1}$$

Prenons l'exemple de la taille en centimètres d'une personne (ou d'un objet observé). Supposons que nous voulions définir l'ensemble des personnes de « taille moyenne ». En logique classique, nous conviendrons par exemple que les personnes de taille moyenne sont celles dont la taille est comprise entre 160cm et 180cm. Le sous-ensemble $\mathcal{A}$ des tailles comprises entre 160cm et 180cm a pour fonction caractéristique :

$$\mu_{\mathcal{A}}(x_i) = \begin{cases} 1 & si \quad 160 \leq x_i \leq 180 \\ 0 & sinon \end{cases}$$

La figure 5.1 montre le graphe de cette fonction caractéristique. Elle est binaire et peut être exprimée dans le contexte de l'algèbre booléen, en faisant correspondre le vrai booléen (1) avec l'intervalle $[160, 180]$ et le faux booléen (0) avec l'union d'intervalles $[0, 160] \cup [180, +\infty]$.

5.2.1.2 Théorie des ensembles flous

Dans la théorie classique des ensembles, un élément ne peut appartenir à la fois à un ensemble et à son complémentaire ; il ne peut pas, non plus, n'appartenir à aucun des deux. Cela signifie qu'un élément x_i est soit dans $\mathcal{A}$ ($\mu_{\mathcal{A}}(x_i) = 1$) ou non ($\mu_{\mathcal{A}}(x_i) = 0$). Or dans plusieurs situations, il est parfois ambigu que x_i appartienne ou non à $\mathcal{A}$. Il existe en effet beaucoup de concepts pour lesquels nous ne trouvons pas de frontières bien définies, par exemple, la jeunesse, la proximité, la maturité d'un fruit. La logique floue permet de représenter ces notions aux limites imprécises par des ensembles dont le degré d'appartenance n'est plus binaire mais graduel ; permettre des

2. En traitement d'images, $\mathcal{X}$ sera typiquement l'espace dans lequel est définie l'image ($\mathbb{Z}^n$ ou $\mathbb{R}^n$, avec $n = 2, 3, \ldots$). Les éléments de $\mathcal{X}$ sont alors les points de l'image (pixels, voxels). L'univers peut également être un ensemble de valeurs prises par des caractéristiques de l'image telles que l'échelle des niveaux de gris. Les éléments sont alors des valeurs (niveaux de gris). L'ensemble X peut aussi être un ensemble de primitives ou objets extraits des images (segments, régions, objets, etc.) dans une représentation de plus haut niveau du contenu de l'image.

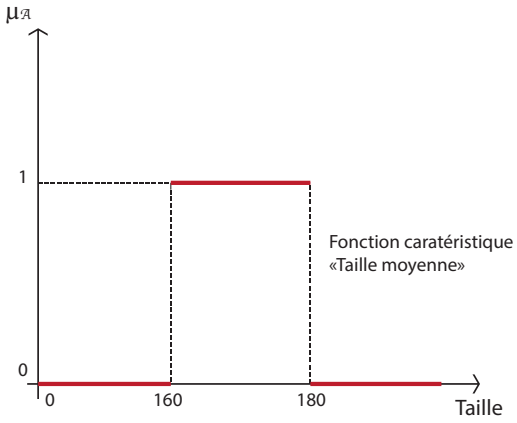


FIGURE 5.1: Fonction caractéristique du sous-ensemble classique $\mathcal{A}$ des tailles comprises entre 160cm et 180cm.

graduations dans l'appartenance d'un élément à un sous-ensemble, c'est-à-dire d'autoriser un élément à appartenir plus ou moins fortement à un sous-ensemble.

Définition 5.2.2. Un *sous-ensemble flou* $\mathcal{A}$ sur un domaine $\mathcal{X}$ est décrit en utilisant différentes notations. Il peut être noté comme l'ensemble des couples :

$$\mathcal{A} = \{(x_i, \mu_{\mathcal{A}}(x_i)) \,|x_i \in \mathcal{X}\} \tag{5.2.2}$$

Il est parfois noté :

$$\mathcal{A} = \int_{\mathcal{X}} \mu_{\mathcal{A}}(x_i)\,/x_i \tag{5.2.3}$$

ou dans le cas discret :

$$\mathcal{A} = \sum_{i=1}^{N} \mu_{\mathcal{A}}(x_i)\,/x_i \tag{5.2.4}$$

où N désigne le cardinal de $\mathcal{X}$. Remarquez que le signe « / » n'est pas la division mais un séparateur, et que la « somme » représente l'union des éléments dans un ensemble. La fonction d'appartenance $\mu_{\mathcal{A}} : \mathcal{X} \to [0,1]$ associe à chaque élément x_i de $\mathcal{X}$ un réel entre 0 et 1 (l'intervalle $[0,1]$ est ici pris comme définition car il est le plus employé, d'autres intervalles ou ensembles peuvent cependant être utilisés, tels que les treillis). Pour les valeurs 0 et 1, nous somme ramenés à l'appartenance et à la non appartenance classique. La situation la plus floue et la plus difficile est celle où $\mu_A(x_i) = 0{,}5$ [Klir et al., 1997].

$$\mu_{\mathcal{A}}(x_i) = \begin{cases} \mu_{\mathcal{A}}(x_i) = 1 & x_i \text{ appartient entièrement à } \mathcal{A} \\ 0 \leq \mu_{\mathcal{A}}(x_i) \leq 1 & x_i \text{ appartient partiellement à } \mathcal{A} \\ \mu_{\mathcal{A}}(x_i) = 0 & x_i \text{ n'appartient pas à } \mathcal{A} \end{cases} \tag{5.2.5}$$

Supposons vouloir caractériser l'ensemble $\mathcal{B}$ des personnes avoisinant les 170cm, la proposition avoisinant les 170cm n'est pas précise. $\mathcal{B}$ pourrait être représenté par une fonction caractéristique, mais faut-il choisir les tailles comprises entre 160cm et 180cm ou 165cm et 175cm comme étant celles avoisinant 170cm. La fonction d'appartenance $\mu_{\mathcal{B}}$ possible pour $\mathcal{B}$ est illustrée sur la figure 5.2.

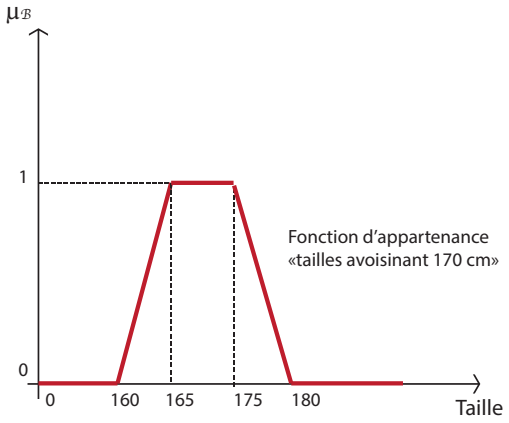


FIGURE 5.2: Fonction d'appartenance du sous-ensemble flou $\mathcal{B}$ des tailles avoisinant 170cm.

5.2.2 Entropie de Shannon

Le principe de l'entropie, un concept bien connu de la théorie de l'information et introduit par Shannon [Cover & Thomas, 1991], est d'utiliser l'incertitude comme une unité de mesure pour décrire l'information contenue dans une source. L'information maximale est atteinte lorsque aucune connaissance *a priori* n'est disponible, et dans ce cas, il en résulte une incertitude maximale.

Considérons une variable aléatoire X qui peut prendre une valeur parmi N possibles dans un alphabet $\mathcal{S}$ de N éléments. La probabilité de prendre une valeur $s_i \in \mathcal{S}$ est p_i. Nous tirons au hasard une valeur s_i. Plus cette valeur est improbable, plus nous sommes « surpris » de la voir apparaître. Le fait que nous soyons surpris est clairement lié au fait que nous ne sommes pas *certain* du résultat de l'expérience avant sa réalisation, même si nous pouvons décrire l'ensemble de tous les résultats possibles. La question que nous nous posons ici est :« *Comment pourrions-nous mesurer cette incertitude* ? » Intuitivement, nous avons une incertitude maximale si nous ne connaissons pas les probabilités, ou si nous posons des probabilités uniformes (égales) sur toutes les possibilités. Connaissant les probabilités *a priori* de chacun des évènements qui caractérisent les résultats de l'expérience, il est déjà possible de prévoir que l'incertitude associée à des évènements équiprobables est supérieure à celle associée au cas où il y a des évènements qui ont une probabilité d'occurrence plus grande que d'autres, ou au cas extrême où l'un des évènement est certain ($p_i = 1$). Dans ce dernier cas, l'incertitude associée à l'expérience est *nulle*, c'est-à-dire minimale.

Définition 5.2.3. Considérons un ensemble fini $\mathcal{S} = \{s_1, s_2, \ldots, s_N\}$ d'évènements indépendants de probabilité d'occurrence p_1, p_2, …, p_N. La mesure d'incertitude, appelée **entropie de Shannon**, associée à l'ensemble $\mathcal{S}$ est calculée par :

$$H(p_1, p_2, \ldots, p_N) = -\sum_{i=1}^{N} p_i \log_2 p_i \tag{5.2.6}$$

avec $\sum_{i=1}^{N} p_i = 1$.

Le symbole $\log_2$ désigne le logarithme en base 2. En choisissant la base 2, l'expression de l'entropie utilise l'incertitude d'une expérience ayant deux évènements équiprobables ($\mathcal{S} = \{s_1, s_2\}$ où $p_1 = p_2 = 0{,}5$) comme unité de mesure. L'entropie s'exprime ainsi en bit (*binary unit*) [3]. Les bits représentent les réalisations possibles de la variable aléatoire X. Notons que le choix de la base du logarithme a peu d'importance, car il est possible de passer d'une base à une autre à l'aide de $\log_a b = \log_a c . \log_c b$.

Nous présentons ci-après quelques propriétés associées à la fonction d'entropie. Les preuves peuvent être trouvées dans [Guiasu & Theodorescu, 1971].

i L'entropie est maximale lorsque les événements sont équiprobables, $p_i = \frac{1}{N}$ et vaut alors $\log(N)$; c'est dans cette configuration que le système est le moins bien défini et que les messages sont les plus « surprenants » et apportent le plus d'information.

ii L'information est minimale lorsque l'un des événements est certain : le système est parfaitement connu, et aucun apport d'information n'est possible. $H(p_1, p_2, \ldots, p_N) = 0$ si $\exists i | p_i = 1$.

iii pour N événements équiprobables, l'entropie croît avec N.

iv l'entropie est une fonction positive. $H(p_1, p_2, \ldots, p_N) \geq 0$

5.2.3 L'entropie floue

Une mesure du flou, aussi appelée entropie floue (*Fuzzy Entropy*) ou entropie non probabiliste représentant le degré de flou, est une extension de l'entropie probabiliste de Shannon [Shannon, 1948] aux évènements flous. De Luca et Termini [De Luca & Termini, 1972] ont proposé un indice d'entropie non probabiliste, ou indice de flou (*fuzziness*), basé sur la satisfaction d'un ensemble d'axiomes. Une fonction qui utilise l'entropie probabiliste de Shannon a été proposée dans le cadre de ces axiomes.

Définition 5.2.4. Supposons $\mathcal{C}$ l'ensemble des ensembles nets de $\mathcal{X}$, et $\mathcal{F}$ l'ensemble d'ensembles flous dans $\mathcal{X}$. Soient $\mathcal{A}$ et $\mathcal{B}$ des sous-ensembles, flous ou non, d'un même univers $\mathcal{X}$. Définissons l'application $E : \mathcal{F} \to [0, 1]$. E est un indice d'entropie non probabiliste, ou indice de flou si elle satisfait les quatre axiomes de De Luca et Termini [De Luca & Termini, 1972] :

1. $E(\mathcal{A}) = 0$ si et seulement si $\mathcal{A} \in \mathcal{C}$ (les ensembles nets sont complètement non flous et sont les seuls à vérifier cette propriété).

3. Il ne faut pas confondre le bit (*binary unit*), ou Shannon, avec le bit de l'informatique (*binary digit*), et qui est simplement un chiffre binaire.

2. $E(\mathcal{A}) = 1$ (valeur maximale) si et seulement si $\mu_{\mathcal{A}}(x) = 0{,}5$, $\forall x \in \mathcal{X}$.
3. $E(\mathcal{A}) \leq E(\mathcal{B})$ si $A \leq_S B$. La relation d'ordre $\leq_S$ est un opérateur de comparaison appelé *sharpeness*. Un ensemble flou $\mathcal{A}$ est considéré plus *sharpened* que $\mathcal{B}$ si :

$$\begin{array}{llll} \mu_{\mathcal{A}}(x) \leq \mu_{\mathcal{B}}(x) & si & \mu_{\mathcal{B}}(x) \leq 0{,}5 \\ \mu_{\mathcal{A}}(x) \geq \mu_{\mathcal{B}}(x) & si & \mu_{\mathcal{B}}(x) \geq 0{,}5 \end{array} \tag{5.2.7}$$

.

4. $E(\mathcal{A}) = E(\mathcal{A}^c)$, avec $\mathcal{A}^c$ est le complementaire de l'ensemble $\mathcal{A}$.

De Luca et Termini ont également défini l'entropie d'un ensemble flou pour le calcul de l'indice de *fuzziness*. Cette entropie est basée sur l'entropie de Shannon et elle satisfait tous les axiomes du degré de flou. Elle est donnée par [De Luca & Termini, 1972] :

$$H(\mathcal{A}) = -K \sum_{x_i \in \mathcal{X}} \mathcal{S}(\mu_{\mathcal{A}}(x_i)) \tag{5.2.8}$$

où A est l'ensemble flou, $K \in \mathbb{R}^+$ est une constante de normalisation et $\mathcal{S}$ est l'entropie probabiliste de Shannon, donnée par :

$$\mathcal{S}(\mu_{\mathcal{A}}(x_i)) = \mu_{\mathcal{A}}(x_i) \log \mu_{\mathcal{A}}(x_i) + (1 - \mu_{\mathcal{A}}(x_i)) \log (1 - \mu_{\mathcal{A}}(x_i)) \tag{5.2.9}$$

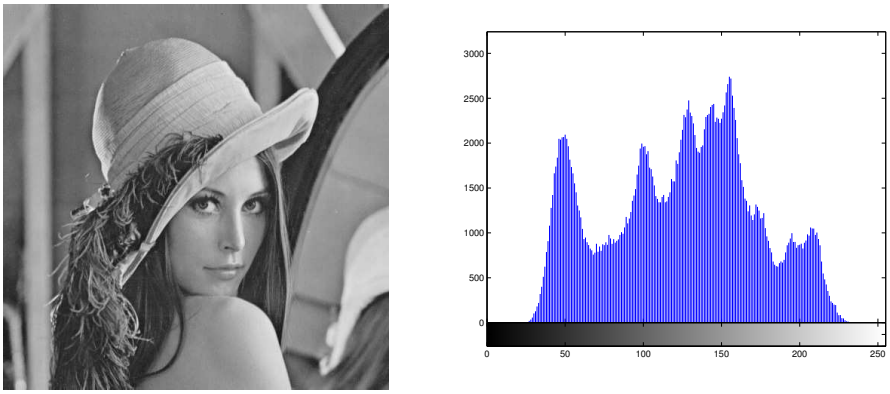
Les indices de *fuzziness* sont des éléments largement utilisés dans le domaine de la gestion de l'information et de l'incertitude inhérentes aux systèmes complexes. Un certain nombre d'études portant sur les mesures d'entropie floue et ses applications ont été menées par Kaufmann [Kaufmann, 1975], Bhandari & Pal [Bhandari & Pal, 1993], Kapur [Kapur, 1997], Pal et al. [Pal & Pal, 1992], et Fan et al. [Fan & Xie, 1999].

5.2.4 La représentation d'une image

La notion d'image qui est utilisée dans la suite est de nature bidimensionnelle discrète. Elle est représentée par une matrice de M lignes et N colonnes. Chaque élément de la matrice est appelé un *pixel* (*picture element*). Le pixel est désigné par un couple (x, y). A chaque pixel de l'image, correspond un niveau d'intensité lumineuse, appelé niveau de gris (*grey level*) appartenant à $\{0, 1, \cdots, L-1\}$ où 0 correspond au manque total d'illumination (couleur noire), $L - 1$ est la couleur blanche et les autres valeurs sont des niveaux de gris entre le noir et le blanc [Braviano, 1995].

Définition 5.2.5. Soit $G = \{0, 1, \ldots, L-1\}$ l'ensemble des niveaux de gris de l'image, et $D = \{(x,y) : 0 \leq x \leq M-1, 0 \leq y \leq N-1\}$ les coordonnées spatiales des pixels pour une image $M \times N$. On appelle image une application $I : D \longrightarrow G$, qui à tout point $p = (x,y)$ de l'espace bidimensionnel fini D fait correspondre une luminance ou niveau de gris $I(x,y)$, avec $0 \leq I(x,y) \leq L-1$. En général, on considère que les images sont codées sur 256 niveaux de gris pour des images codées sur 8 bits.

Pour une image monochrome codée sur L niveaux, on définit l'*histogramme des niveaux de gris* comme étant une fonction discrète de $\{0,\ldots,L-1\}$ dans $\mathbb{N}$, qui associe à chaque niveau de gris entre 0 et $L-1$ la quantité de pixels de l'image qui possèdent cette intensité lumineuse. La figure 5.3 présente un exemple d'histogramme d'une image.



(a) (b)

FIGURE 5.3: Exemple d'histogramme de l'image « Lena »

On utilise aussi souvent l'*histogramme normalisé* qui est une fonction de $\{0,\ldots,L-1\}$ dans [0,1].

Définition 5.2.6. On définit $D_k = \{(x,y)|I(x,y)=k,(x,y)\in D\}$, l'ensemble des pixels qui ont la valeur k, $k \in G$. L'*histogramme normalisé* d'une image de hauteur M et de largeur N, défini comme $\mathcal{H} = \{h_0, h_1, \ldots, h_{L-1}\}$, présente la fréquence d'apparition des différents niveaux de gris qui composent l'image. Le niveau k dans l'image est défini comme :

$$h_k = \frac{n_k}{N * M} \qquad (k = 0,1,\ldots,L-1) \tag{5.2.10}$$

où n_k est le nombre d'occurrences du niveau k (nombre total de pixels dans D_k), et $N * M$ est le nombre total de pixels de l'image.

Il est clair que :

$$0 \leq h_k \leq 1 \quad et \quad \sum_{k=0}^{L-1} h_k = 1 \tag{5.2.11}$$

La partition probabiliste (PP : *probability partition*) d'une image dans le domaine D est définie comme

$$\Pi_L = \{D_0, D_1, \ldots, D_{L-1}\}$$

et elle est caractérisée par la distribution de probabilité [Tao et al., 2003, 2007] :

$$p_k \equiv P(D_k) = h_k, \quad k = 0,1,\ldots,L-1\ , \tag{5.2.12}$$

L'équation (5.2.12) présente la relation entre l'histogramme $\mathcal{H}$ et la *partition probabiliste* Π_L, où p_k est la probabilité d'occurrence d'un pixel de niveau k. L'histogramme peut alors être

vu comme une densité de probabilité qui fournit la probabilité de trouver un certain niveau de gris de l'image. Ainsi le niveau de gris d'un pixel devient une variable aléatoire dont la valeur dépend du résultat d'une expérience aléatoire sous-jacente. D'où un traitement statistique des images.

5.2.5 Segmentation

En traitement d'image, on dit qu'on a segmenté une image lorsqu'on a partitionné l'espace où elle est définie en zones homogènes au regard d'un critère que l'on s'est donné.

Définition 5.2.7. La segmentation d'une image numérique consiste à partitionner l'espace des pixels d'une image I en un ensemble de régions R_i homogènes au sens d'un certain critère. Il s'agit donc d'une fonction $\xi : D \longrightarrow \mathcal{R}$ qui associe à chaque pixel p l'indice i de la région R_i à laquelle il appartient. Chaque région R_i est construite comme une composante connexe, c'est-à-dire un ensemble de pixels adjacents de valeur i, ayant des propriétés communes qui les différencient des pixels des régions voisines. Plus formellement, Zucker [Zucker, 1976] définit une segmentation de l'ensemble des pixels d'une image I, comme l'ensemble des régions R_i $(i = 1, \ldots, n)$, tel que :

1. $\forall i$, $R_i \neq \varnothing$;
2. $\forall i \neq j$, $R_i \cap R_j = \varnothing$;
3. $I = \cup_i R_i$;
4. R_i est connexe $\forall i$;
5. il existe un prédicat $\mathcal{P}$ tel que $\mathcal{P}(R_i) =$ vrai, $\forall i$;
6. $\mathcal{P}(R_i \cup R_j) =$ faux si $i \neq j$ et R_i et R_j sont adjacentes.

où $\varnothing$ dénote l'ensemble vide. Le prédicat $\mathcal{P}$ est un critère d'homogénéité qui peut être fondé sur la luminance, ou un indice de texture, par exemple. Ce prédicat est vrai si R_i est homogène, faux dans le cas contraire.

Cette définition ne conduit pas à une méthode de segmentation unique et il peut exister plusieurs segmentations possibles d'une image avec l'utilisation d'un même prédicat.

Les méthodes de segmentation existantes peuvent être séparées en deux approches couramment qualifiées d'approches « contours » (*edge-based techniques*) et d'approche « régions » (*region-based segmentation*). La segmentation par contours est basée sur la recherche des discontinuités locales présentes dans l'image, tandis que dans la segmentation par régions, les pixels de chaque région sont connectés par l'uniformité de leurs caractéristiques, telles que la luminance, la couleur, les coordonnées, etc. Les deux approches sont complémentaires et aucune n'a prouvé sa supériorité par rapport à l'autre, chacune ayant ses avantages et ses domaines d'application. La voie est plutôt dans les approches coopératives, qui utilisent conjointement

les deux types de méthodes. Dans ce travail, nous ne nous intéressons qu'à l'approche région et plus précisément aux méthodes de seuillage.

L'ensemble des méthodes de segmentation basées sur les régions peut être décomposé en trois classes. La première concerne les méthodes par division et fusion (*split and merge*). La seconde fait référence à la croissance de régions (*region growing*) et la troisième classe est la segmentation fondée sur la classification ou le seuillage des pixels en fonction de leur intensité (*thresholding*).

5.2.5.1 Approche par division et fusion

Les approches de segmentation par division considèrent l'image comme une seule région. Elle sera ensuite divisée d'une manière récursive, en régions de plus en plus petites, tant que le critère d'homogénéité n'est pas vérifié. Cette phase de division est réalisée selon une structure géométrique. Citons par exemple la structure *quadtree*, où l'image est découpée récursivement en carrés jusqu'à ce que chaque carré soit homogène, la structure de *Delaunay* qui consiste à partitionner l'image en triangles, et la structure de *Voronoï*, qui divise l'image en polygones qui comportent un nombre de côtés non fixe [Braviano, 1995].

À la fin de la procédure de division vient l'étape de fusion, où les régions connexes et similaires sont ou non fusionnées selon un critère de regroupement préétabli, permettant la détection des entités dans l'image.

5.2.5.2 Croissance de régions

La croissance de régions consiste à fusionner un ensemble de pixels ayant des caractéristiques proches. Elle peut se faire à partir de plusieurs niveaux, le plus élémentaires étant le pixel. On distingue entre les méthodes qui regroupent des pixels adjacents de façon itérative et celles qui fusionnent de petites régions selon un critère de fusion basé sur les caractéristiques des régions adjacentes [Gonzalez & Woods, 1992].

Agrégation de pixels La méthode d'agrégation de pixels est une méthode ascendante qui permet de regrouper un ensemble de pixels selon un double critère d'homogénéité et d'adjacence. Elle part d'un ensemble de pixels, judicieusement choisis, appelés « germes ». Chaque germe fusionne avec un premier pixel connexe qui possède des caractéristiques similaires, puis avec un deuxième, et ainsi itérativement de sorte que chaque région croit pixel par pixel.

Regroupement d'ensembles de pixels Au départ, de petites régions-germes sont générées (ces régions-germes peuvent se réduire à un pixel). Un graphe d'adjacence de régions est construit de manière à ce que chaque région-germe soit représentée par un nœud et une

arête est créée entre deux régions adjacentes. Un coût de fusion est associé à chaque arête. La méthode consiste alors à déterminer, à partir de ce nouveau graphe pondéré, les meilleures fusions de façon à minimiser la perte d'information. Cette fusion entraîne une contraction dans le graphe d'adjacence.

5.2.5.3 Les méthodes de classification

La segmentation ou partition d'une image en régions peut être obtenue à l'aide d'une classification appliquée sur l'ensemble des valeurs des pixels. De très nombreuses méthodes de classification existent [Pal & Pal, 1993]. Nous traitons dans ce chapitre de la méthode de segmentation par seuillage, qui est une méthode simple et très populaire pour la segmentation d'objets dans les images numériques. Le seuillage est une méthode de classification supervisée, c'est-à-dire que le nombre de classes 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 problème de seuillage consiste donc à diviser l'image en K classes non forcément connexes, qui vont diviser l'histogramme en K zones. La technique la plus simple procède par binarisation, où les pixels de l'image sont partagés par un seul seuil T en deux classes. Mais il est souvent nécessaire de diviser l'image en un nombre de classes supérieur à deux. Dans ce cas, pour obtenir K classes il faut $K-1$ seuils ; c'est le processus de *multiseuillage (multithresholding).*

Définition 5.2.8. La segmentation par seuillage est une opération qui consiste à répartir les pixels en K classes $(\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_K)$, à partir d'un ensemble de seuils $(T_1, T_2, \ldots, T_{K-1})$, auxquels nous ajoutons les seuils T_0 et T_K qui sont les bornes des différentes régions en niveaux de gris, à savoir 0 et $L-1$ respectivement. Le seuillage peut être représentée comme une fonction qui associe à chaque pixel l'indice k de la classe C_k à laquelle il appartient. Ainsi un pixel aux coordonnées (x, y) d'intensité $I(x,y)$ est affecté à la classe $\mathcal{C}_k$ si $T_k \leq I(x, y) \leq T_{k+1}$, $k \in \{0, \ldots, K-1\}$. Le but d'une méthode de seuillage étant de trouver les seuils optimaux $T_1, \ldots, T_K - 1$.

D'une manière générale, le choix de seuils d'histogramme peut se faire de deux manières différentes. Dans le premier procédé, les seuils sont déterminés globalement pour tous les points de l'image, tandis que dans le second procédé, les seuils sont calculés localement à partir de portions de l'image. Une étude exhaustive sur les méthodes de segmentation par seuillage peut être trouvée dans [Sezgin & Sankur, 2004].

5.3 Présentation du problème

Dans la section précédente, nous avons présenté les briques de base pour la compréhension du problème étudié. Nous présentons tout d'abord le contexte de notre travail, qui, rappelons-le se trouve à la croisée de trois domaines : la segmentation d'image, la théorie des ensembles flous et les métaheuristiques. Nous allons étudier la façon dont la logique floue peut être utilisée pour la segmentation d'image. Une attention particulière sera accordée à l'entropie floue. Une question fondamentale qui se pose est alors de savoir : « *pourquoi recourir à la théorie des ensembles flous pour la segmentation d'image* ? » Pour répondre à cette première question, nous donnons ci-après quelques explications et présentons des approches de la littérature basées sur la notion d'entropie.

Les données d'images, de même que les connaissances nécessaires à l'interprétation, sont le plus souvent entachées d'incertitude et d'imprécision. Cette imperfection trouve ses origines à différents niveaux : phénomènes observés, limites des capteurs et acquisitions des données, bruit, processus numériques de reconstruction, nature des images et mode de représentation de leurs éléments constitutifs, etc. Face aux imperfections relatives aux images, nous pouvons choisir plusieurs attitudes. Une première attitude consiste à éliminer les imperfections autant que possible. Cela passe par exemple par l'amélioration des capteurs et la multiplication des acquisitions. Une autre attitude consiste à doter le système de moyen permettant de représenter l'approximation et de la propager au cours du processus d'interprétation. Trois approches sont le plus souvent utilisées. Les principales sont la *théorie des probabilités*, associés à la théorie bayésienne de la décision, la *théorie des fonctions de croyances* (ou *théorie de Dempster-Shafer* [Shafer, 1976]), qui combine information incomplète et probabilité par utilisation d'ensembles aléatoires et la *théorie des sous-ensembles flous* associées à la théorie des possibilités [Zadeh, 1965].

Si l'on dispose d'une connaissance sur les distributions des niveaux de gris des pixels pour les classes que l'on recherche (en particulier la probabilité d'occurrence d'un niveau de gris pour chaque classe, et la probabilité à priori des classes), alors nous pouvons nous replacer dans les conditions de la théorie bayésienne de la décision et déterminer théoriquement les seuils optimaux qui minimisent le coût des fausses erreurs en se basant sur les règles de Bayes. Cependant, en pratique cela est loin d'être le cas. Pour appréhender alors les informations incertaines et imprécises en analyse d'image et notamment aux applications concernant la segmentation d'image, plusieurs méthodes basées sur le modèle flou [4] ont été développées ces dernières années. L'incertitude dans une image peut s'exprimer soit en termes d'ambiguïté d'appartenance d'un pixel

4. En logique floue, où un élément peut appartenir à plusieurs classes en même temps, la condition 2 (cf. définition 5.2.7) référente à la partition de l'espace des pixels d'une image en sous-ensembles n'est pas, en général, vérifiée.

à l'image ou au fond (s'il est noir ou blanc), soit au niveau de l'indéfinition de la forme et de la géométrie d'une région dans l'image, soit l'association des deux facteurs précédents. Pour évaluer l'ambiguïté associée aux niveaux de gris présents dans une image, il est possible d'utiliser la mesure d'information d'entropie, adaptée au cas flou ; l'incertitude étant de nature floue et non pas de nature aléatoire. Dans les méthodes de segmentation par seuillage, le but est de trouver le(s) meilleur(s) seuil(s) permettant d'extraire les objets de l'image. En raisonnant avec la logique floue, l'objectif est de trouver le(s) seuil(s) qui minimise(ent) l'incertitude associée à l'image. Cette incertitude peut être déterminée par l'indice de flou présenté dans la section 5.2.3.

Parmi la multitude de techniques de seuillage d'images, les approches basées sur l'entropie ont été largement abordées dans la littérature. Fondamentalement, le seuillage entropique (*entropy thresholding*) considère l'histogramme de l'image comme une distribution de probabilité, et détermine les seuils optimaux de manière à maximiser l'entropie résultant du découpage de l'histogramme $\mathcal{H}$ en plusieurs classes. Plus précisément, la meilleure image seuillée est celle qui préserve l'information autant que possible en termes d'entropie de Shannon [Chang et al., 2006].

Le critère populaire pour le seuillage basé sur le principe du maximum d'entropie a d'abord été appliqué par Pun [Pun, 1980, 1981], puis amélioré dans [Kapur et al., 1985]. Le concept a été généralisé par la suite pour évoluer à l'entropie de Renyi [Sahoo et al., 1997] et l'entropie de Tsallis [Tsallis, 1988]. Cheng et al. [Cheng et al., 1998] ont introduit la notion de c-partition floue (*fuzzy c-partition*) dans le principe du maximum d'entropie pour sélectionner les valeurs des seuils pour des images en niveaux de gris. Cette méthode a été appliquée au seuillage à deux niveaux puis étendu au seuillage multi-niveaux. Tobias et Serra [Tobias & Seara, 2002] ont proposé une approche de seuillage d'histogramme déterminée en fonction de la similitude entre les niveaux de gris, et évaluée par l'application d'une mesure floue. Dans [Zhao et al., 2001], Zhao et al. ont proposé une nouvelle technique pour le seuillage à trois niveaux, en exploitant la relation entre la *fuzzy c-partition* et la *partition probabiliste*. Dans la technique proposée, le principe de l'entropie maximale a été utilisé pour mesurer la compatibilité entre la partition floue et la *partition probabiliste*. Les auteurs ont utilisé la fonction la plus simple, qui est de nature monotone, pour l'approximation de l'appartenance des ensembles flous sombre (*dark*), moyen (*medium*) et clair (*bright*) (définis en fonction des niveaux d'intensité lumineuse des pixels) et ont dérivé une condition nécessaire pour la maximisation de la fonction d'entropie. Basé sur l'idée de Zhao et al., Tao et al. [Tao et al., 2003] ont conçu une nouvelle méthode de seuillage à trois niveaux pour la segmentation d'image. Les auteurs ont défini un nouveau concept d'entropie floue à travers une analyse de probabilité, la partition floue et la théorie

de l'entropie. L'image est d'abord divisée en trois parties, à savoir sombre, moyenne et claire, dont les fonctions d'appartenance de la région floue sont décrites par les fonctions Z, Π et S, respectivement. L'approche présentée dans [Tao et al., 2003] est adoptée comme support de base pour ce travail.

5.3.1 Formulation du problème

Nous nous intéressons dans ce travail au cas du seuillage à trois niveaux (*Three-level Thresholding*) d'une image, où le but est de séparer l'espace des pixels D en trois parties, E_d (*dark*), E_m (*medium*) et E_b(*bright*). Le niveau intermédiaire E_m représente les pixels dont les niveaux de gris représentent des valeurs moyennes, E_d est composé de pixels « sombres » correspondant à des valeurs de luminance faibles proches du niveau 0 (noir), et E_b est composé de pixels « clairs » qui représentent des valeurs élevées proches du niveau 255 (blanc). Le seuillage repose sur l'hypothèse que les différents objets de l'image sont différentiables uniquement par leurs intensités lumineuses. 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.

Le problème consiste à trouver la partition probabiliste floue (*3-partition*) de D, $\Pi_3 = \{E_d, E_m, E_b\}$, caractérisée par les distributions de probabilité suivantes [Tao et al., 2003, 2007] :

$$p_d = P(E_d), p_m = P(E_m), p_b = P(E_b) \tag{5.3.1}$$

Les trois partitions floues E_d, E_m et E_b sont caractérisées par trois fonctions d'appartenance μ_d, μ_m et μ_b, respectivement. Pour calculer les degrés d'appartenance des pixels aux différentes classes, trois fonctions sont utilisées. La fonction d'appartenance des pixels sombres μ_d correspond à la forme en Z, la fonction d'appartenance μ_m des pixels moyens est une fonction en Π, et la fonction d'appartenance μ_b des pixels clairs de l'image est une fonction en S [Tao et al., 2003] La figure 5.4 illustre ces trois fonctions. Six paramètres libres $\{a_1, b_1, c_1, a_2, b_2, c_2\}$ contrôlent les formes des trois fonctions d'appartenance et satisfont les conditions $0 < a_1 \leq b_1 \leq c_1 \leq a_2 \leq b_2 \leq c_2 < 255$ pour une image de 256 niveaux de gris.

Pour séparer les niveaux de gris en trois classes, il faut trouver deux seuils optimaux T_1 et T_2 qui partagent l'histogramme de ces niveaux de façon à retenir un maximum d'information. La classification d'un pixel (x, y) est réalisée comme suit :

$$\begin{aligned} D_{kd} &= \{(x,y) : I(x,y) \leq T_1, (x,y) \in D_k\} \\ D_{km} &= \{(x,y) : T_1 < I(x,y) \leq T_2, (x,y) \in D_k\} \\ D_{kb} &= \{(x,y) : I(x,y) > T_2, (x,y) \in D_k\} \end{aligned} \tag{5.3.2}$$

$\Pi_k = \{D_{kd}, D_{km}, D_{kb}\}$ est la *partition probabiliste* de D_k (ensemble des pixels qui ont la valeur k ; $k \in G$) avec la distribution de probabilité suivante :

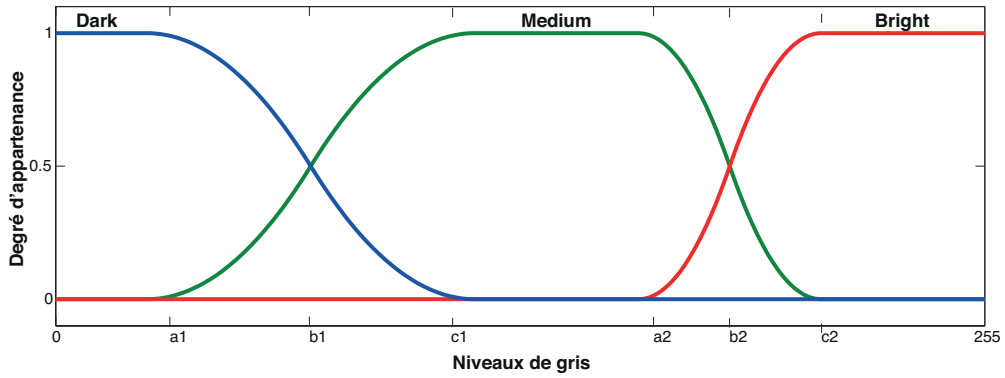


FIGURE 5.4: Graphe des fonctions d'appartenance

$$\begin{aligned} p_{kd} &= P\left(D_{kd}\right) = p_k.p_{d|k} \\ p_{km} &= P\left(D_{km}\right) = p_k.p_{m|k} \\ p_{kb} &= P\left(D_{kb}\right) = p_k.p_{b|k} \end{aligned} \tag{5.3.3}$$

Dans l'équation (5.3.3), p_k est la distribution de probabilité de la partition probabiliste (cf. équation (5.2.12)) ; $p_{d|k}$ est la probabilité conditionnelle d'être en présence de la classe d (sombre) sachant que le pixel appartient à D_k. De même, $p_{m|k}$ et $p_{b|k}$ sont les probabilités conditionnelles d'un pixel appartenant aux classes m (moyenne) et b (claire), respectivement. Selon la formule des probabilités totales, nous avons donc :

$$\begin{aligned} p_d &= \sum_{k=0}^{255} p_k.p_{d|k} = \sum_{k=0}^{255} p_k.\mu_d\left(k\right) \\ p_m &= \sum_{k=0}^{255} p_k.p_{m|k} = \sum_{k=0}^{255} p_k.\mu_m\left(k\right) \\ p_b &= \sum_{k=0}^{255} p_k.p_{b|k} = \sum_{k=0}^{255} p_k.\mu_b\left(k\right) \end{aligned} \tag{5.3.4}$$

Il est clair que le problème de seuillage à trois niveaux se résume à trouver les fonctions d'appartenance appropriés $\mu_d\left(k\right)$, $\mu_m\left(k\right)$ et $\mu_b\left(k\right)$ d'un pixel avec un niveau d'intensité arbitraire k. Ces fonctions d'appartenance représentent la probabilité conditionnelle qu'un pixel, d'un niveau de gris $k \in G$, appartienne à l'une des classes sombre, moyenne ou claire, respectivement (i.e., $p_{d|k} = \mu_d\left(k\right)$, $p_{m|k} = \mu_m\left(k\right)$ et $p_{b|k} = \mu_b\left(k\right)$). La partition probabiliste floue considère que pour chaque point, la somme des mesures d'appartenance pour toutes les classes vaut 1. Ainsi, $\mu_d\left(k\right) + \mu_m\left(k\right) + \mu_b\left(k\right) = 1$, $k = 0, 1, \ldots, 255$.

Les trois fonctions représentant les degrés d'appartenance de chaque niveau de gris, présent dans l'image, à chacune des classes, sont présentées dans les équations (5.3.5), (5.3.6) et (5.3.7).

La fonction d'appartenance $\mu_d(k)$ en forme de Z est de type décroissante, entièrement définie par la donnée des paramètres a_1, b_1 et c_1. Elle est définie par :

$$\mu_d(k) = \begin{cases} 1 & k \leq a_1 \\ 1 - \frac{(k-a_1)^2}{(c_1-a_1)*(b_1-a_1)} & a_1 < k \leq b_1 \\ \frac{(k-c_1)^2}{(c_1-a_1)*(c_1-b_1)} & b_1 < k \leq c_1 \\ 0 & k > c_1 \end{cases} \tag{5.3.5}$$

La fonction en Π associe un pixel à une plage de niveaux de gris de valeur moyenne. À chaque valeur de croisement possible avec les deux autres fonctions correspond une partition floue de l'image. La fonction d'appartenance $\mu_m(k)$ est donnée par :

$$\mu_m(k) = \begin{cases} 0 & k \leq a_1 \\ \frac{(k-a_1)^2}{(c_1-a_1)*(b_1-a_1)} & a_1 < k \leq b_1 \\ 1 - \frac{(k-c_1)^2}{(c_1-a_1)*(c_1-b_1)} & b_1 < k \leq c_1 \\ 1 & c_1 < k \leq a_2 \\ 1 - \frac{(k-a_2)^2}{(c_2-a_2)*(b_2-a_2)} & a_2 < k \leq b_2 \\ \frac{(k-c_2)^2}{(c_2-a_2)*(c_2-b_2)} & b_2 < k \leq c_2 \\ 0 & k > c_2 \end{cases} \tag{5.3.6}$$

La fonction d'appartenance $\mu_b(k)$ en forme de S est définie par :

$$\mu_b(k) = \begin{cases} 0 & k \leq a_2 \\ \frac{(k-a_2)^2}{(c_2-a_2)*(b_2-a_2)} & a_2 < k \leq b_2 \\ 1 - \frac{(k-c_2)^2}{(c_2-a_2)*(c_2-b_2)} & b_2 < k \leq c_2 \\ 1 & k > c_2 \end{cases} \tag{5.3.7}$$

Afin de tenir compte de l'information ambiguë fournie par les niveaux de gris proches de la frontière séparatrice des trois classes, nous utilisons la mesure d'information définie par l'entropie floue. Les paramètres libres des trois fonctions d'appartenance peuvent être déterminés en maximisant l'entropie de la partition floue [Tao et al., 2003, 2007]. L'entropie totale de la partition Π_3 est définie comme suit :

$$H\left(a_1, b_1, c_1, a_2, b_2, c_2\right) = H_d + H_m + H_b \tag{5.3.8}$$

où,

$$\begin{aligned} H_d &= -\sum_{k=0}^{255} \frac{p_k * \mu_d(k)}{p_d} * \ln\left(\frac{p_k * \mu_d(k)}{p_d}\right) \\ H_m &= -\sum_{k=0}^{255} \frac{p_k * \mu_m(k)}{p_m} * \ln\left(\frac{p_k * \mu_m(k)}{p_m}\right) \\ H_b &= -\sum_{k=0}^{255} \frac{p_k * \mu_b(k)}{p_b} * \ln\left(\frac{p_k * \mu_b(k)}{p_b}\right) \end{aligned} \tag{5.3.9}$$

Les seuils optimaux T_1 et T_2 qui séparent l'image en trois classes de niveaux de gris correspondent aux points de croisement des fonctions d'appartenance pour lesquels l'ambiguïté est maximale, c'est-à-dire :

$$\mu_d(T_1) = \mu_m(T_1) = 0{,}5 \tag{5.3.10}$$

$$\mu_m(T_2) = \mu_b(T_2) = 0{,}5 \tag{5.3.11}$$

Sur la base des équations (5.3.5), (5.3.6) et (5.3.7), nous pouvons écrire [Tao et al., 2003] :

$$T_1 = \begin{cases} a_1 + \sqrt{(c_1 - a_1) * (b_1 - a_1)/2} & \text{si } (a_1 + c_1)/2 \leq b_1 \leq c_1 \\ c_1 - \sqrt{(c_1 - a_1) * (c_1 - b_1)/2} & \text{si } a_1 \leq b_1 < (a_1 + c_1)/2 \end{cases} \tag{5.3.12}$$

$$T_2 = \begin{cases} a_2 + \sqrt{(c_2 - a_2) * (b_2 - a_2)/2} & \text{si } (a_2 + c_2)/2 \leq b_2 \leq c_2 \\ c_2 - \sqrt{(c_2 - a_2) * (c_2 - b_2)/2} & \text{si } a_2 \leq b_2 < (a_2 + c_2)/2 \end{cases} \tag{5.3.13}$$

Comme mentionné précédemment, pour trouver la combinaison optimale de tous les paramètres flous, nous proposons d'utiliser une nouvelle technique d'optimisation basée sur l'hybridation des algorithmes BBO et DE.

5.4 Principe de la méthode DBBO-Fuzzy

Vu les bonnes capacités d'exploration de l'algorithme DE [Noman & Iba, 2008], une méthode hybride combinant l'exploitation de BBO à l'exploration de DE est proposée dans le présent chapitre. Le but de cette hybridation est de bénéficier des avantages de chaque algorithme. La méthode proposée, nommée DBBO-Fuzzy, emploie la mutation de l'algorithme DE. En plus, une procédure de sélection est introduite de manière à favoriser un nombre donné d'individus pour la génération suivante. L'algorithme proposé peut se résumer comme suit.

5.4.1 Initialisation

L'algorithme génère aléatoirement un ensemble de vecteurs de variables en faisant des tirages uniformes dans chacun des domaines associés aux composantes de l'espace de recherche. Une solution potentielle au problème à résoudre est représentée par un vecteur de dimension D. Chaque composante de ce vecteur désigne un paramètre flou. Résoudre le problème de seuillage à trois niveau exposé plus haut, consiste à déterminer l'ensemble de ces paramètres $\{a_1, b_1, c_1, a_2, b_2, c_2\}$ maximisant l'entropie totale floue. Le nombre de ces paramètre correspond à la dimension du problème, soit $(D = 6)$. La notation suivante est adoptée pour représenter le $i^{\text{ème}}$ individu de la population à la génération g :

$$\vec{X}_{i,g} = (X_{i,1,g}, X_{i,2,g}, X_{i,j,g}, \ldots, X_{i,D,g}) \tag{5.4.1}$$

où $i = 1, \ldots, NP$, $j = 1, \ldots, D$ et NP la taille de la population. Chaque variable de décision, $X_{i,j,g}$, est initialisée aléatoirement, de façon uniforme, à l'intérieur de la borne inférieure (L_j) et la borne supérieure (U_j) de l'espace de recherche, de la manière suivante :

$$X_{i,j,0} = L_j + rand(0,1) \times (U_j - L_j) \tag{5.4.2}$$

5.4.2 Évaluation de la fonction objectif

Dans la section 5.3.1, nous avons présenté le critère permettant d'évaluer la qualité d'un seuillage à trois niveaux. Ce critère se base sur le principe de maximisation de l'entropie totale floue. Rappelons ici la formulation de la fonction objectif :

$$H\,(a_1, b_1, c_1, a_2, b_2, c_2) = H_d + H_m + H_b \tag{5.4.3}$$

où H_d, H_m et H_b sont les entropies partielles des classes *sombre*, *moyenne* et *claire* respectivement (cf. équation (5.3.10)). Les seuils optimaux T_1 et T_2 qui séparent l'image en trois classes de niveaux de gris sont calculés sur la base des paramètres flous $(a_1, b_1, c_1, a_2, b_2, c_2)$ (équations (5.3.12) et (5.3.13)), dont il est question de trouver la séquence optimale qui maximise l'entropie totale H et qui satisfait la condition $0 < a_1 \leq b_1 \leq c_1 \leq a_2 \leq b_2 \leq c_2 < 255$.

5.4.3 Migration

L'opérateur de migration produit une nouvelle solution, notée $\vec{M}_{i,g}$, comme suit :

$$M_{i,j,g} = \begin{cases} X_{k,j,g} & \text{si } rand(0,1) < \lambda_i \\ X_{i,j,g} & \text{sinon} \end{cases} \tag{5.4.4}$$

où $i = 1, 2, \ldots, NP$, $j = 1, \ldots, D$ et $X_{k,j,g}$ est la $j^{\text{ème}}$ variable de décision de l'individu $\vec{X}_{k,g}$. Cet individu est choisi avec une probabilité proportionnelle à son taux d'émigration μ_k.

5.4.4 Mutation

La mutation produit pour chaque solution $\vec{M}_{i,g}$, issue de l'étape précédente, un mutant $\vec{V}_{i,g}$ suivant le schéma *DE/rand/1/bin* :

$$\vec{V}_{i,g} = \vec{M}_{r_1,g} + F(\vec{M}_{r_2,g} - \vec{M}_{r_3,g}) \tag{5.4.5}$$

où les indices r_1, r_2 et r_3 sont choisis au hasard dans l'intervalle $[1, NP]$ et doivent être mutuellement différents de l'indice courant i ; $F \in [0, 2]$ est le cœfficient de mutation.

5.4.5 Sélection

Un individu $\vec{X}_{i,g}$ est remplacé par un individu mutant $\vec{V}_{i,g}$, si ce dernier est meilleur que lui. Ainsi, la survie d'un individu sera directement reliée à sa performance au sein de la population. Cela traduit bien l'idée de la sélection naturelle. La règle de sélection est la suivante :

$$\vec{X}_{i,g+1} = \begin{cases} \vec{V}_{i,g} & \text{si } f(\vec{V}_{i,g}) < f(\vec{X}_{i,g}) \\ \vec{X}_{i,g} & \text{si } f(\vec{V}_{i,g}) > f(\vec{X}_{i,g}) \end{cases} \tag{5.4.6}$$

5.4.6 Respecter les bornes de l'espace de recherche

Si une solution est générée en dehors de l'espace de recherche, elle est corrigée de la façon suivante :

$$X_{i,j,g} = L_j + rand(0, 1) \times (U_j - L_j) \tag{5.4.7}$$

5.4.7 Élitisme

La stratégie élitiste consiste à conserver les n_{elit} meilleurs individus à chaque génération. Après chaque évaluation de la performance des individus à une génération g donnée, les n_{elit} meilleurs individus de la génération précédente $(g - 1)$ sont réintroduits dans la population pour remplacer les n_{elit} individus les moins performants.

5.4.8 Critère d'arrêt

Le critère d'arrêt de l'algorithme est défini en fonction du nombre maximal absolu de générations ; passé ce nombre de générations, l'algorithme est terminé.

5.5 Expérimentation numérique et résultats

Cette section a pour but d'appliquer l'algorithme DBBO-Fuzzy à la segmentation d'image. Notre but est de montrer que l'algorithme proposé peut être utilisé dans le cas de la segmentation d'image. Nous ne cherchons pas à trouver des images mieux segmentées que celles proposées

dans les articles de la littérature, ni de faire une comparaison poussée avec les techniques existantes. De même, nous n'avons pas cherché à optimiser notre algorithme pour la segmentation d'image. Une étude plus poussée sur cette adaptation pourrait être réalisée plus tard.

La performance de l'algorithme DBBO-Fuzzy est comparée à celles de l'algorithme de base BBO [Simon, 2008], nommé ici comme BBO-Fuzzy, et les performances de DE-Fuzzy, qui se déroule exactement comme l'algorithme original présenté dans [Price et al., 2005].

5.5.1 Images de test

Nous avons choisi, pour comparer les performances des algorithmes proposés, une série de douze images de référence. Ces images sont communément connues sous le nom de *Lena, Peppers, Cameraman, Airplane, Lake, Walking bridge, Mandrill, Barbara, Boat, Elaine, GoldHill* et *Fingerprint*. Les douze images sont représentées en 256 niveaux de gris et leur taille est de 256×256 pixels. Nous présentons les images originales et les histogrammes correspondants dans les figures 5.5 et 5.6.

5.5.2 Paramétrage

Le tableau 5.1 récapitule la liste des paramètres de l'algorithme DBBO-Fuzzy et leur réglage. Nous avons fixé la taille de la population NP à 20. Pour une comparaison équitable avec les autres algorithmes en compétition, le même nombre d'individus est utilisé. Les paramètres de l'algorithme DE-Fuzzy sont : $F = 0{,}5$ et $CR = 0{,}9$, comme recommandé dans [Storn & Price, 1997]. Le même schéma de mutation, *DE/rand/1/bin*, est adopté à la fois pour le DE-Fuzzy et DBBO-Fuzzy. Les paramètres de l'algorithme BBO-Fuzzy sont les mêmes que ceux utilisés dans [Simon, 2008], excepté la taille de la population qui est fixé à 20.

Pour chaque image de test, nous avons effectué dix exécutions indépendantes de chaque algorithme, durant un nombre de générations $\max_{gen} = 20$.

Paramètres	Notation	Valeur
Taille de la population	NP	20
Paramètre d'élitisme	n_{elit}	2
Taux maximum d'immigration	I	1
Taux maximum d'émigration	E	1
Nombre de générations	$\max_{gen}$	20
Domaine de définition des variables	$[L_i, U_i]$	$[0, 255]$
Dimension du problème (nombre de paramètres flous)	D	6
Constante de mutation	F	0,5
Schéma de mutation	*DE/rand/1/bin*	

TABLEAU 5.1: Valeurs des paramètre de l'algorithme DBBO-Fuzzy

Afin d'isoler l'effet de chaque paramètre sur les performance de DBBO-Fuzzy, des essais ont

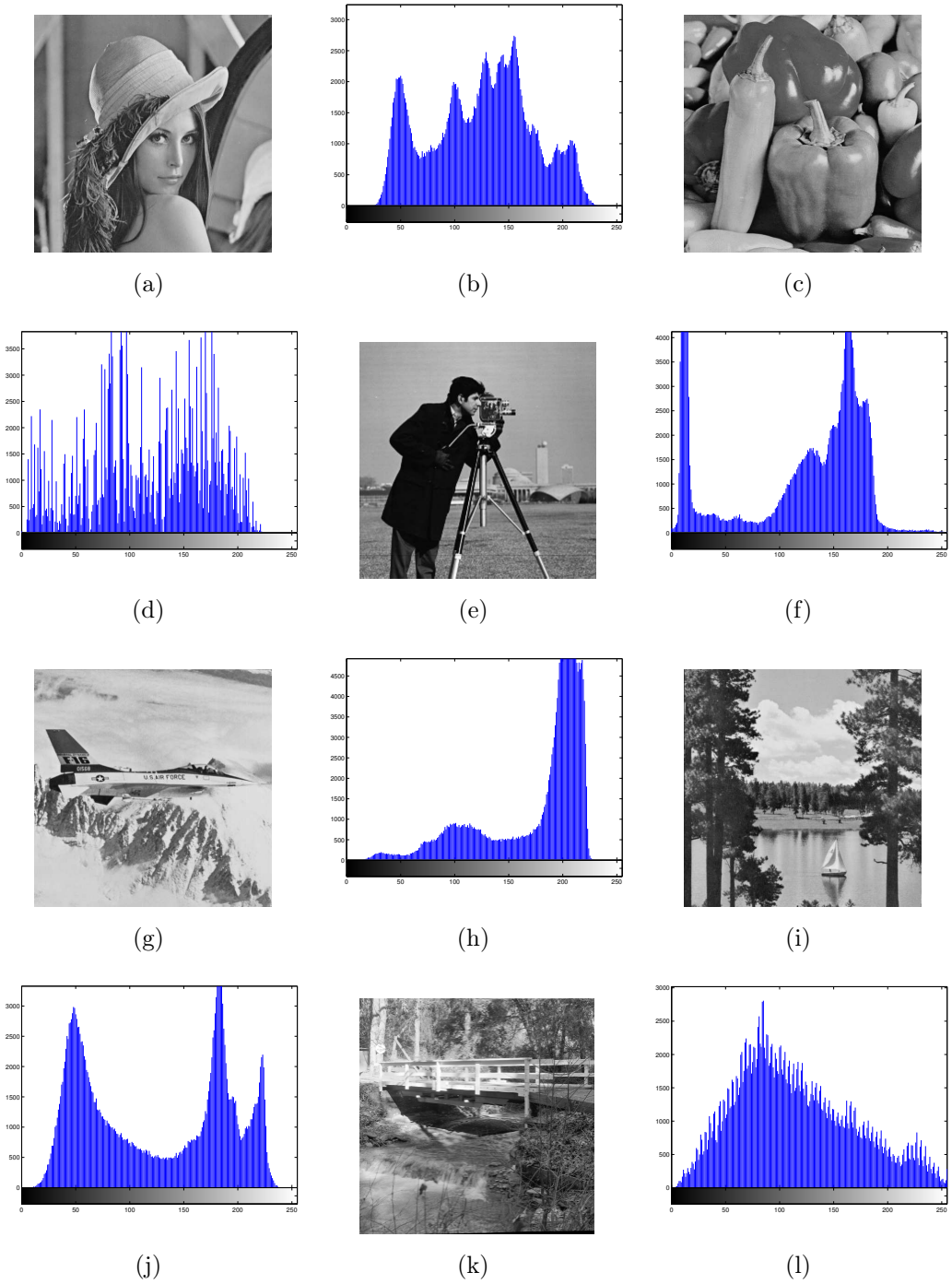


FIGURE 5.5: Images de test originales et leurs histogrammes : (a) Lena, (b) Histogramme de Lena, (c) Peppers, (d) Histogramme de Peppers, (e) Cameraman, (f) Histogramme de Cameraman, (g) Airplane, (h) Histogramme de Airplane, (i) Lake, (j) Histogramme de Lake, (k) Walking bridge, (l) Histogramme de Walking bridge

été faits en n'en faisant varier qu'un seul à la fois. Ainsi la section 5.5.3.1, présente l'effet de la variation du schéma de mutation, la section 5.5.3.2 illustre l'effet de la variation de la taille de la population et enfin, la section 5.5.3.3 montre l'effet de la variation du paramètre d'élitisme.

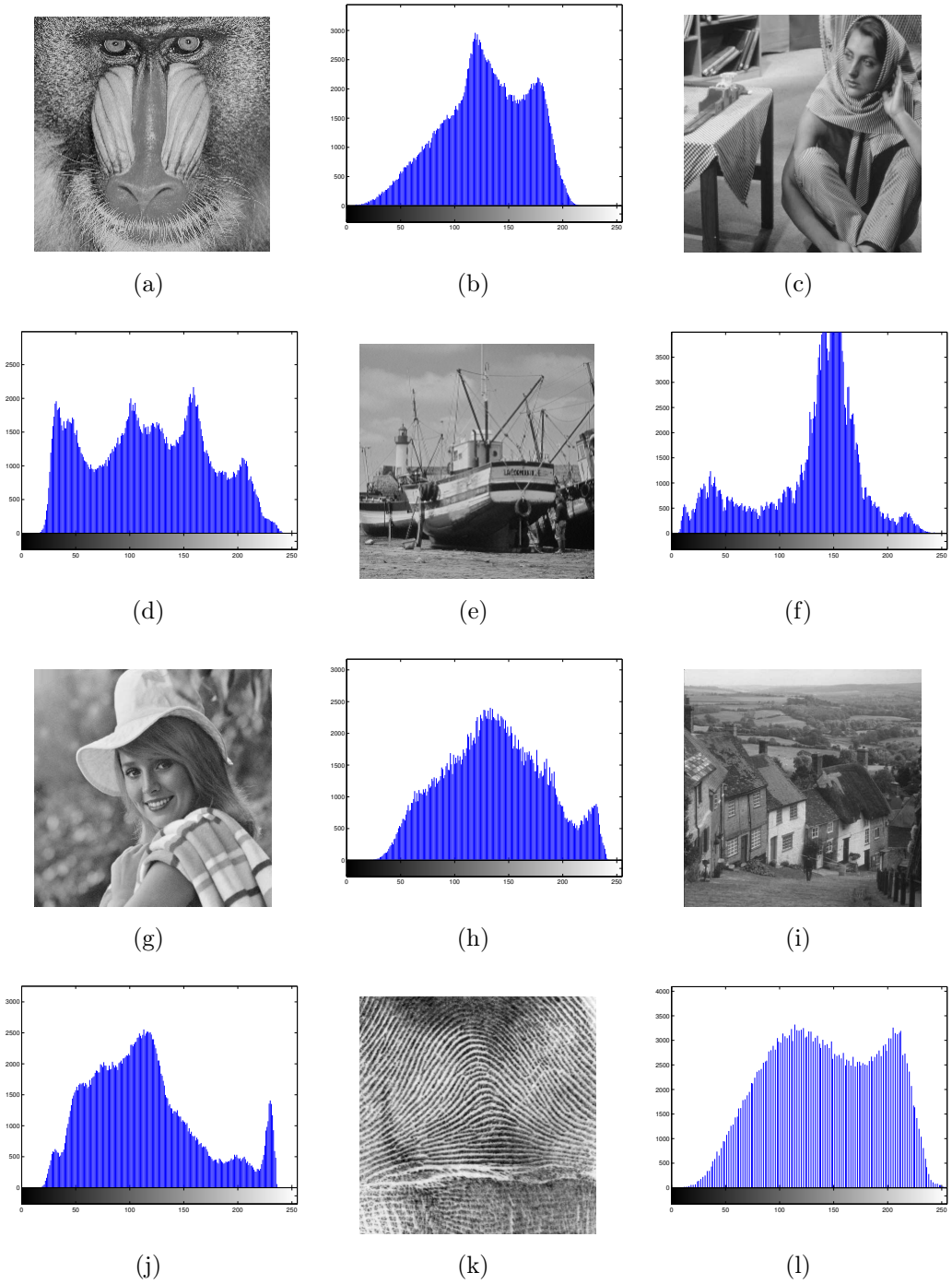


FIGURE 5.6: (Cont.)Images de test originales et leurs histogrammes : (a) Mandrill, (b) Histogramme de Mandrill, (c) Barbara, (d) Histogramme de Barbara, (e) Boat, (f) Histogramme de Boat,(g) Elaine, (h) Histogramme de Elaine, (i) GoldHill, (j) Histogramme de GoldHill, (k) Fingerprint, (l) Histogramme de Fingerprint

5.5.3 Résultats et Discussions

Nous présentons maintenant les résultats expérimentaux obtenus avec les méthodes proposées. Les résultats de segmentation sont présentés à la figure 5.7.

Les algorithmes de seuillage à trois niveaux sont utilisés pour déterminer les seuils optimaux T_1 et T_2 qui divisent l'image en trois niveaux de gris tout en préservant les informations d'origine autant que possible après la création de cette partition. L'objectif est de trouver la combinaison « optimal » de tous les paramètres flous $(a_1, b_1, c_1, a_2, b_2, c_2)$ qui maximise l'entropie totale floue $H(b_1, c_1, a_2, b_2, c_2)$, donnée par l'équation (5.3.8). Plus grande est la valeur de la fonction objectif des résultats, meilleure est la segmentation.

Comme il est difficile d'estimer visuellement la qualité des résultats d'une procédure de segmentation, nous proposons d'utiliser des critères quantitatifs. Différentes méthodes d'évaluation quantitative de la segmentation existent dans la littérature [Zhang, 1996]. Elles peuvent être groupées en deux catégories, selon que l'on possède ou non une vérité terrain ou une segmentation de référence. Quand on dispose d'une vérité terrain, l'évaluation des segmentations est basée sur une mesure de similarité entre l'image segmentée et sa vérité terrain. En l'absence de vérité terrain, l'évaluation de la qualité d'une segmentation se base sur des statistiques calculées sur les régions.

Le critère d'uniformité intra-région de Levine et Nazif [Levine & Nazif, 1985], que nous avons retenu pour l'évaluation de la qualité des résultats, ne nécessite pas de vérité terrain. Il est donné par :

$$u = 1 - 2 * c * \frac{\sum_{j=0}^{c} \sum_{j \in R_j} (f_i - \mu_j)^2}{N * (f_{max} - f_{min})^2} \tag{5.5.1}$$

où,

- c : nombre de seuils ;
- R_j : $j^{\text{ème}}$ région segmentée ;
- f_i : niveau de gris du pixel i ;
- μ_j : moyenne des niveaux de gris des pixels de la région R_j ;
- N : nombre total des pixels dans l'image ;
- f_{max} : valeur maximale des niveaux de gris ;
- f_{min} : valeur minimale des niveaux de gris.

La valeur de cette mesure d'uniformité doit être dans l'intervalle [0,1]. Plus la valeur de u est importante, plus le résultat de la segmentation est considéré comme étant satisfaisant.

Le tableau 5.2 indique, pour chaque image de test, la valeur moyenne de la *fitness* (entropie totale de la partition floue) obtenue sur les différentes exécutions et la déviation (écart type). Les valeurs en *gras* indiquent l'algorithme le plus performant parmi les algorithmes concurrents. Il est observé, à partir des résultats obtenus, que l'algorithme DBBO-Fuzzy obtient des valeurs d'entropie supérieures à celles obtenues par les deux autres algorithmes, pour toutes les images de test.

Afin de déterminer si les moyennes calculées sont significativement différentes d'un point de vue statistique, entre l'algorithme DBBO-Fuzzy et les deux autres algorithmes, un *test de Student* a été réalisé avec un degré de liberté $d.d.l. = 10 + 10 - 2 = 18$ et un seuil de signification $\alpha = 0{,}05$ (c'est-à-dire 5% de chances de rejeter par erreur l'hypothèse nulle, soit un niveau de confiance de 95%). La valeur absolue calculée de t étant, pour toutes les images de test, au-delà du seuil critique, nous pouvons rejeter l'hypothèse nulle d'égalité des moyennes et en conclure que les moyennes obtenues par DBBO-Fuzzy et les deux autres algorithmes sont significativement différents d'un point de vue statistique. Par conséquent, il est évident que l'hybridation de l'algorithme BBO avec DE a un effet notable sur la performance de chacun des algorithmes.

Images	Valeurs moyennes de la *fitness*			Écart type			1 vs 2	1 vs 3
	DBBO-Fuzzy	BBO-Fuzzy	DE-Fuzzy	DBBO-Fuzzy	BBO-Fuzzy	DE-Fuzzy	test t	test t
Lena	**1,3851E+01**	1,3569E+01	1,3707E+01	5,3148E-02	2,8230E-02	4,1078E-02	1,4823E+01 †	6,7622E+00 †
Peppers	**1,3443E+01**	1,3089E+01	1,3314E+01	3,4273E-02	5,6806E-02	5,8473E-02	1,6883E+01 †	6,0430E+00 †
Cameraman	**1,3463E+01**	1,3090E+01	1,3235E+01	3,8302E-02	8,7798E-02	5,8679E-02	1,2307E+01 †	1,0301E+01 †
Airplane	**1,3543E+01**	1,3380E+01	1,3451E+01	2,5830E-02	6,9561E-02	5,9266E-02	6,9381E+00 †	4,4722E+00 †
Lake	**1,3817E+01**	1,3641E+01	1,3739E+01	2,2140E-02	8,0443E-02	3,2497E-02	6,6555E+00 †	6,2791E+00 †
Walk Bridge	**1,4550E+01**	1,4193E+01	1,4441E+01	4,3054E-02	5,7350E-02	2,9707E-02	1,5728E+01 †	6,5527E+00 †
Mandrill	**1,3815E+01**	1,3593E+01	1,3725E+01	4,1507E-02	4,7428E-02	3,621E-02	1,1112E+01 †	5,1733E+00 †
Barbara	**1,4214E+01**	1,3845E+01	1,4097E+01	5,7206E-02	6,9849E-02	8,0261E-02	1,2925E+01 †	3,7587E+00 †
Boat	**1,3986E+01**	1,3596E+01	1,3880E+01	8,9041E-02	7,0177E-02	7,3576E-02	1,0883E+01 †	2,9037E+00 †
Elaine	**1,4089E+01**	1,3771E+01	1,3999E+01	3,5250E-02	6,0926E-02	4,9027E-02	1,4276E+01 †	4,7264E+00 †
GoldHill	**1,4069E+01**	1,3693E+01	1,3920E+01	6,7739E-02	3,7019E-02	4,5656E-02	1,5410E+01 †	5,7761E+00 †
Fingerprint	**1,2406E+01**	1,2100E+01	1,2328E+01	4,1010E-02	5,8752E-02	4,7234E-02	1,3486E+01 †	3,9108E+00 †

TABLEAU 5.2: Comparaison des performances de DBBO-Fuzzy, BBO-Fuzzy et DE-Fuzzy, où les valeurs en **gras** indiquent le meilleur algorithme. Les résultats sont les meilleurs moyennes de la fitness et les écarts types sur dix exécutions des algorithmes († la différence entre les deux algorithmes est statistiquement significative). Dans ces tests, 1 vs. 2 signifie DBBO-Fuzzy vs. BBO-Fuzzy, et 1 vs. 3 signifie DBBO-Fuzzy vs. DE-Fuzzy.

Le tableau 5.3 indique les seuils optimaux et les valeurs du facteur d'uniformité calculées selon l'équation (5.5.1), obtenus par les trois algorithmes. Nous pouvons observer à partir des résultats, que la qualité de la solution de DBBO-Fuzzy est supérieure à celle de BBO-Fuzzy et DE-Fuzzy pour 11 images sur un total de 12. De ce point de vue aussi, l'algorithme DBBO-Fuzzy s'impose comme le grand vainqueur.

Le tableau 5.4 présente l'ensemble des paramètres optimaux représentatifs $\{a_1, b_1, c_1, a_2, b_2, c_2\}$ obtenus par les différents algorithmes en compétition.

Pour une interprétation visuelle des résultats de seuillage à trois niveaux, les images obtenues par application de l'algorithme DBBO-Fuzzy sont présentées dans la figure 5.7. Après détermination des seuils pour chaque image, les niveaux de gris de tous les pixels dans une région donnée, sont remplacés par la valeur moyenne des niveaux de gris des pixels de la région considérée.

Images	Seuils optimaux			Mesure d'uniformité		
	DBBO-Fuzzy	BBO-Fuzzy	DE-Fuzzy	DBBO-Fuzzy	BBO-Fuzzy	DE-Fuzzy
Lena	98, 175	98, 169	83, 161	**9,8391E-01**	9,8246E-01	9,7213E-01
Peppers	85, 162	71, 205	63, 188	**9,7720E-01**	9,6821E-01	9,6464E-01
Cameraman	127, 207	132, 211	97, 198	9,6557E-01	9,5745E-01	**9,6767E-01**
Airplane	41, 166	46, 178	30, 153	**9,8719E-01**	9,8666E-01	9,8618E-01
Lake	96, 171	95, 165	64, 161	**9,8451E-01**	9,8420E-01	9,7778E-01
Walk Bridge	79, 154	35, 211	48, 180	**9,7816E-01**	9,6507E-01	9,6149E-01
Mandrill	74, 150	33, 143	51,167	**9,6755E-01**	9,6617E-01	9,6448E-01
Barbara	93, 170	93, 172	90, 170	**9,8233E-01**	9,7595E-01	9,7342E-01
Boat	107, 185	111, 220	75, 217	**9,8327E-01**	9,7904E-01	9,7566E-01
Elaine	98, 175	37, 208	44, 196	**9,7517E-01**	9,6634E-01	9,6595E-01
GoldHill	77, 152	52, 156	67, 181	**9,8172E-01**	9,7873E-01	9,7072E-01
Fingerprint	94, 172	41, 152	42, 179	**9,7856E-01**	9,6897E-01	9,5562E-01

TABLEAU 5.3: Seuils optimaux et valeurs du facteur d'uniformité obtenus par les algorithmes DBBO-Fuzzy, BBO-Fuzzy et DE-Fuzzy, où les valeurs en **gras** indiquent le meilleur algorithme

Images	Paramètres optimaux		
	DBBO-Fuzzy	BBO-Fuzzy	DE-Fuzzy
Lena	1, 137, 140, 140, 142, 256	5, 136, 138, 141, 144, 232	1, 115, 118, 119, 124, 256
Peppers	1, 119, 122, 122, 124, 256	5, 98, 99, 104, 246, 248	1, 83, 95, 107, 194, 256
Cameraman	1, 177, 180, 180, 194, 256	28, 174, 175, 179, 215, 235	82, 94, 116, 119, 211, 256
Airplane	1, 1, 136, 138, 138, 233	5, 7, 141, 147, 149, 250	1, 1, 100, 101, 132, 240
Lake	1, 135, 136, 136, 136, 256	7, 131, 131, 132, 138, 239	20, 77, 87, 95, 143, 256
Walk Bridge	1, 111, 111, 111, 112, 256	1, 3, 116, 118, 245, 254	1, 26, 131, 141, 156, 256
Mandrill	1, 103, 105, 105, 106, 256	1 18 90 94 102 250	1, 40, 120, 128, 131, 256
Barbara	1, 131, 132, 133, 137, 256	2, 124, 137, 140, 146, 242	23, 115, 120, 134, 134, 256
Boat	1, 151, 151, 151, 161, 256	24, 143, 150, 151, 245, 253	1, 100, 113, 124, 256, 256
Elaine	1, 136, 140, 141, 143, 256	2, 5, 117, 118, 244, 246	1, 1, 149, 152, 187, 256
GoldHill	1, 108, 108, 109, 110, 256	20, 27, 119, 120, 121, 241	1, 89, 100, 121, 173, 256
Fingerprint	1, 132, 132, 135, 138, 256	4, 25, 105, 106, 114, 252	1, 1, 142, 145, 149, 256

TABLEAU 5.4: Ensembles représentatifs des paramètres optimaux $(a_1, b_1, c_1, a_2, b_2, c_2)$

La figure 5.8 montre quelques exemples d'images seuillées après application de chacun des algorithmes. Par souci de concision, nous ne présentons qu'une partie des images. Sur cette figure, les images originales sont illustrés en 5.8(a), 5.8(e), 5.8(i), 5.8(m), et en 5.8(q). Les images segmentées (en 3 classes) sont illustrées en 5.8(b), 5.8(f), 5.8(j), 5.8(n), et en 5.8(r) pour DBBO-Fuzzy, en 5.8(c), 5.8(g), 5.8(k), 5.8(o), et en 5.8(s) pour BBO-Fuzzy et en 5.8(d), 5.8(h), 5.8(l), 5.8(p), et en 5.8(t) pour DE-Fuzzy. Une interprétation visuelle des résultats de seuillage montre clairement la supériorité de DBBO-Fuzzy pour les images retenues.

5.5.3.1 Effet de la stratégie de mutation

Le but de cette section est d'explorer de manière plus approfondie, l'effet de la stratégie de mutation sur les performances de DBBO-Fuzzy. Les résultats sont rapportés dans tableau 5.5. Pour mettre en évidence la meilleure stratégie, les meilleures valeurs sont données en **gras**.

(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

(j) (k) (l)

FIGURE 5.7: Seuillage à trois niveaux des images en utilisant DBBO-Fuzzy : (a) Lena, (b) Peppers, (c) Cameraman, (d) Airplane, (e) Lake, (f) Walking bridge, (g) Mandrill, (h) Barbara, (i) Boat, (j) Elaine, (k) GoldHill, (l) Fingerprint

Un examen plus attentif du tableau 5.5 révèle que, sur les douze images de test, l'algorithme DBBO-Fuzzy avec la stratégie de mutation *DE/best/1* donne de meilleurs résultats. En effet,

l'entropie est maximale pour huit images de test, à savoir : *Lena, Peppers, Cameraman, Airplane, Lake, Mandrill, Elaine* et *Fingerprint*. L'algorithme DBBO-Fuzzy avec *DE/rand/2* s'est avéré être meilleur dans deux cas seulement (*Walking bridge* et *Boat*). Les deux stratégies de mutation *DE/rand/1* et *DE/current-to-best/1* donnent de meilleurs résultats dans un seul cas, *GoldHill* et *Barbara* respectivement.

Pour ce qui est de la mesure d'uniformité, DBBO-Fuzzy avec *DE/rand/1* produit les meilleures valeurs pour six images sur les douze (*Lena, Cameraman, Walking bridge, Mandrill, Boat* et *Fingerprin*). Pour les six autres images de test, DBBO-Fuzzy avec le schéma de mutation *DE/current-to-best/1*, sort le vainqueur.

Un *test de Student* a été conduit avec un seuil de signification fixé à 0,05. Ce test a pour but de vérifier si la différence entre la variante de l'algorithme DBBO-Fuzzy utilisant le schéma de mutation *DE/rand/1* et les autres variantes est significative. En général, nous pouvons noter que les résultats, en utilisant différentes stratégies de mutation, n'affectent pas significativement les performances de l'algorithme proposé.

5.5.3.2 Effet de la taille de la population

La modification de la taille de la population a-t-elle une très grande influence sur le résultat final. Pour répondre à cette question, des simulations ont été réalisées pour différentes valeurs de taille de la population NP, et les performances de l'algorithme DBBO-Fuzzy sont présentées pour les différentes variations dans la figure 5.9.

En général, nous pouvons déduire que l'algorithme DBBO-Fuzzy avec une taille de la population $NP = 20$ se montre le meilleur comparé aux autres variantes, car il atteint avec cette configuration la valeur maximale de l'entropie floue dans onze cas sur douze. De même que pour le facteur d'uniformité, où l'algorithme avec une taille de population égale à 20 enregistre les meilleures performances dans huit cas. Ces résultats suggèrent que le fait d'augmenter aveuglément la taille de la population ne peut pas avoir forcément un effet positif sur la performance de l'algorithme.

5.5.3.3 Effet de l'élitisme

Les performances de l'algorithme DBBO-Fuzzy sont également évaluées en détail en faisant varier le paramètre d'élitisme n_{elit}. En général, nous pouvons observer sur la figure 5.10 que les valeurs moyennes de la fonction objectif tendent à augmenter avec le nombre d'élites, bien que la mesure d'uniformité ne suit pas la même tendance pour toutes les images de test. Dans neuf cas sur douze, DBBO-Fuzzy avec $n_{elit} = 8$ produit les meilleurs résultats. De meilleurs résultats sont trouvés dans deux cas seulement pour la variante de l'algorithme avec $n_{elit} = 6$ et dans un

seul cas pour la variante de DBBO-Fuzzy avec $n_{elit} = 4$.

Enfin, une remarque générale importante s'impose ici. S'il y a un choix contradictoire entre une valeur maximale de l'entropie floue et une valeur plus grande du facteur d'uniformité, la priorité devrait être accordée à la solution ayant une plus grande valeur du facteur d'uniformité, car il reflète quantitativement l'impact direct de la qualité de l'image segmentée.

5.6 Conclusion

La segmentation d'image est un processus de partitionnement d'un espace de l'image en plusieurs régions homogènes. Le seuillage est une des techniques les plus largement utilisées dans la segmentation d'image en raison de son application rapide et facile. Cependant, en raison de l'imprécision et l'incertitude inhérentes aux images, le résultat du seuillage n'est pas toujours satisfaisant. Cette incertitude peut être analysée de manière adéquate par l'utilisation de la théorie des ensembles flous, reconnue comme un outil mathématique adéquat qui a prouvé son efficacité et son utilité dans de nombreuses applications, y compris les problèmes de traitement d'images. L'entropie floue est un critère qui permet d'exprimer de façon quantitative un indice de mesure du degré de flou. Les indices de flou sont des éléments largement utilisés avec une grande importance dans le domaine de la gestion de l'information et de l'incertitude dans les systèmes complexes. Dans ce travail, l'entropie floue est utilisée pour déterminer les seuils optimaux d'une image à partir de son histogramme. Les seuils optimaux sont trouvés quand l'entropie est maximale. Le problème de seuillage devient alors un problème d'optimisation qui peut être résolu par n'importe quelle métaheuristique.

Nous avons présenté dans ce chapitre, une approche hybride, DBBO-Fuzzy, permettant de réaliser un seuillage avec un nombre de classes fixé à trois sur des images monochromes. DBBO-Fuzzy a été testé sur différentes images et nous avons pu observer sa robustesse en terme de qualité de solutions trouvées et son efficacité pour déterminer les paramètre optimaux qui maximisent l'entropie. Nous avons ensuite comparé notre méthode aux algorithmes BBO-Fuzzy et DE-Fuzzy. Les résultats expérimentaux montrent que les performances de l'algorithme DBBO-Fuzzy sont meilleures que celles des deux autres algorithmes.

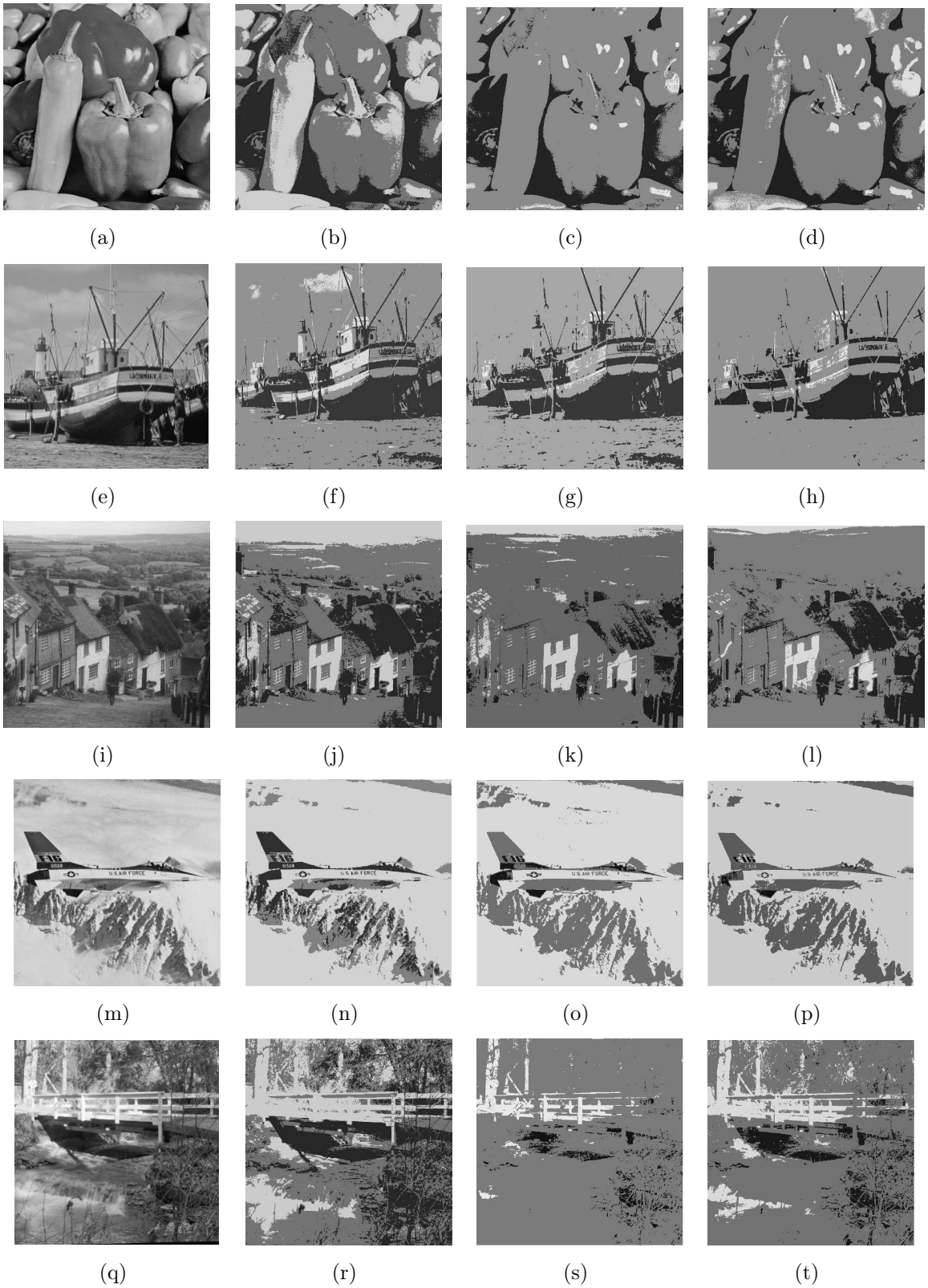


FIGURE 5.8: Comparaison du résultat de seuillage sur un échantillon d'images : (de gauche à droite) la première colonne représente les images originales, la deuxième colonne les images seuillées en utilisant DBBO-Fuzzy, la troisième colonne les images seuillées en utilisant BBO-Fuzzy et la dernière colonne les images seuillées en utilisant DE-Fuzzy

Images	Schémas de Mutation	Seuils optimaux	Moyenne Obj.± Std. Dev	Mesure d'uniformité	Paramètres optimaux
Lena	DE/rand/1	98, 175	1,3851E+01 ± 5,3148E-02	**9,8391E-01**	1, 137, 140, 140, 142, 256
	DE/current-to-best/1	93, 169	1,3847E+01 ± 5,3418E-02	9,8099E-01	1, 131, 132, 132, 133, 256
	DE/best/1	96, 171	**1,3889E+01 ± 6,8730E-02**	9,8222E-01	1, 136, 136, 136, 136, 256
	DE/rand/2	99, 174	1,3815E+01 ± 7,8345E-02	9,8302E-01	1, 139, 140, 140, 141, 256
	DE/best/2	99, 178	1,3764E+01 ± 8,4071E-02 †	9,7977E-01	1, 138, 140, 145, 146, 256
Peppers	DE/rand/1	85, 162	1,3443E+01 ± 3,4273E-02	9,7720E-01	1, 119, 122, 122, 124, 256
	DE/current-to-best/1	86, 162	1,3444E+01 ± 3,9021E-02	**9,7728E-01**	1, 118, 118, 118, 119, 256
	DE/best/1	83, 158	**1,3462E+01 ± 4,0636E-02**	9,7576E-01	1, 117, 117, 117, 117, 256
	DE/rand/2	79, 156	1,3426E+01 ± 5,9953E-02	9,7660E-01	1, 111, 113, 114, 115, 256
	DE/best/2	88, 164	1,3392E+01 ± 5,8353E-02 †	9,7469E-01	1, 124, 125, 126, 126, 256
Cameraman	DE/rand/1	127, 207	1,3463E+01 ± 3,8302E-02	**9,6557E-01**	1, 177, 180, 180, 194, 256
	DE/current-to-best/1	131, 207	1,3431E+01 ± 8,2592E-02	9,6261E-01	1, 184, 185, 185, 187, 256
	DE/best/1	129, 207	**1,3484E+01 ± 7,9968E-02**	9,6267E-01	1, 182, 182, 182, 192, 256
	DE/rand/2	129, 210	1,3391E+01 ± 1,2207E-01	9,6487E-01	1, 182, 183, 184, 196, 256
	DE/best/2	124, 233	1,3365E+01 ± 1,0216E-01 †	9,6514E-01	1, 175, 176, 176, 256, 256
Airplane	DE/rand/1	41, 166	1,3543E+01 ± 2,5830E-02	9,8719E-01	1, 1, 136, 138, 138, 233
	DE/current-to-best/1	104, 170	1,3571E+01 ± 1,8272E-02 †	**9,8772E-01**	1, 146, 147, 147, 147, 227
	DE/best/1	94, 164	**1,3580E+01 ± 2,3179E-02** †	9,8733E-01	1, 133, 133, 133, 134, 239
	DE/rand/2	46, 177	1,3524E+01 ± 3,4368E-02	9,8662E-01	1, 1, 154, 157, 157, 224
	DE/best/2	98, 172	1,3511E+01 ± 1,5145E-02 †	9,8743E-01	1, 138, 139, 140, 153, 234
Lake	DE/rand/1	96, 171	1,3817E+01 ± 2,2140E-02	9,8451E-01	1, 135, 136, 136, 136, 256
	DE/current-to-best/1	98, 172	1,3511E+01 ± 1,5145E-02 †	**9,8743E-01**	1, 138, 139, 140, 153, 234
	DE/best/1	94, 169	**1,3835E+01 ± 2,2273E-02**	9,8431E-01	1, 132, 133, 133, 133, 256
	DE/rand/2	85, 165	1,3775E+01 ± 3,1484E-02 †	9,8342E-01	1, 119, 122, 124, 130, 256
	DE/best/2	98, 175	1,3781E+01 ± 4,6093E-02 †	9,8339E-01	1, 137, 138, 138, 145, 256
Walk Bridge	DE/rand/1	79, 154	1,4550E+01 ± 4,3054E-02	**9,7816E-01**	1, 111, 111, 111, 112, 256
	DE/current-to-best/1	77, 153	1,4551E+01 ± 5,9241E-02	9,7610E-01	1, 109, 109, 110, 111, 256
	DE/best/1	90, 165	1,4555E+01 ± 6,2526E-02	9,7631E-01	1, 127, 127, 127, 127, 256
	DE/rand/2	86, 166	**1,4571E+01 ± 4,2417E-02**	9,7803E-01	1, 118, 125, 126, 130, 256
	DE/best/2	34, 157	1,4502E+01 ± 3,9284E-02 †	9,7543E-01	1, 1, 113, 115, 116, 256
Mandrill	DE/rand/1	74, 150	1,3815E+01 ± 4,1507E-02	**9,6755E-01**	1, 103, 105, 105, 106, 256
	DE/current-to-best/1	31, 147	1,3801E+01 ± 6,0275E-02	9,6600E-01	1, 1, 102, 102, 102, 256
	DE/best/1	29, 144	**1,3845E+01 ± 1,8303E-02**	9,6485E-01	1, 1, 97, 97, 97, 256
	DE/rand/2	30, 145	1,3808E+01 ± 3,3253E-02	9,6669E-01	1, 1, 99, 99, 100, 256
	DE/best/2	36, 161	1,3759E+01 ± 4,6532E-02 †	9,6713E-01	1, 1, 119, 119, 123, 256
Barbara	DE/rand/1	93, 170	1,4214E+01 ± 5,7206E-02	9,8233E-01	1, 131, 132, 133, 137, 256
	DE/current-to-best/1	96, 172	**1,4235E+01 ± 3,3796E-02**	**9,8280E-01**	1, 136, 136, 137, 137, 256
	DE/best/1	104, 178	1,4206E+01 ± 6,4360E-02	9,7831E-01	1, 146, 146, 146, 146, 256
	DE/rand/2	94, 171	1,4228E+01 ± 3,7467E-02	9,8236E-01	1, 131, 134, 135, 137, 256
	DE/best/2	96, 172	1,4154E+01 ± 7,4299E-02	9,7929E-01	1, 134, 137, 137, 138, 256
Boat	DE/rand/1	107, 185	1,3986E+01 ± 8,9041E-02	**9,8327E-01**	1, 151, 151, 151, 161, 256
	DE/current-to-best/1	99, 221	1,4017E+01 ± 6,4776E-02	9,8016E-01	1, 140, 140, 140, 252, 256
	DE/best/1	109, 184	1,4019E+01 ± 8,8207E-02	9,7937E-01	1, 154, 154, 154, 155, 256
	DE/rand/2	100, 223	**1,4022E+01 ± 6,6237E-02**	9,8178E-01	1, 137, 144, 144, 256, 256
	DE/best/2	97, 205	1,3921E+01 ± 8,5438E-02	9,8041E-01	1, 136, 137, 138, 215, 256
Elaine	DE/rand/1	98, 175	1,4089E+01 ± 3,5250E-02	9,7517E-01	1, 136, 140, 141, 143, 256
	DE/current-to-best/1	89, 164	1,4124E+01 ± 5,6691E-02	**9,7828E-01**	1, 126, 126, 126, 127, 256
	DE/best/1	91, 165	**1,4127E+01 ± 6,1915E-02**	9,7573E-01	1, 128, 128, 128, 128, 256
	DE/rand/2	100, 178	1,4107E+01 ± 4,8732E-02	9,7639E-01	1, 139, 144, 146, 146, 256
	DE/best/2	95, 171	1,4078E+01 ± 7,1696E-02	9,7630E-01	1, 133, 134, 136, 136, 256
GoldHill	DE/rand/1	77, 152	**1,4069E+01 ± 6,7739E-02**	9,8172E-01	1, 108, 108, 109, 110, 256
	DE/current-to-best/1	86, 161	1,4037E+01 ± 9,6897E-02	**9,8176E-01**	1, 121, 121, 121, 121, 256
	DE/best/1	94, 169	1,4056E+01 ± 6,5751E-02	9,8152E-01	1, 133, 133, 133, 133, 256
	DE/rand/2	87, 163	1,4027E+01 ± 9,1891E-02	9,8148E-01	1, 120, 124, 124, 125, 256
	DE/best/2	81, 159	1,3991E+01 ± 5,8453E-02 †	9,7927E-01	1, 110, 118, 119, 120, 256
Fingerprint	DE/rand/1	94, 172	1,2406E+01 ± 4,1010E-02	**9,7856E-01**	1, 132, 132, 135, 138, 256
	DE/current-to-best/1	76, 151	1,2389E+01 ± 2,3523E-02	9,7463E-01	1, 107, 108, 108, 108, 256
	DE/best/1	85, 160	**1,2416E+01 ± 3,3792E-02**	9,7398E-01	1, 119, 120, 120, 120, 256
	DE/rand/2	37, 161	1,2396E+01 ± 2,5875E-02	9,7335E-01	1, 2, 121, 121, 122, 256
	DE/best/2	34, 156	1,2336E+01 ± 5,5379E-02 †	9,7322E-01	1, 1, 112, 113, 117, 256

TABLEAU 5.5: Effet de la stratégie de mutation sur les performances de DBBO-Fuzzy. Les meilleures valeurs sont en **gras** († la différence, comparée à la variante de DBBO-Fuzzy utilisant le schéma de mutation *DE/rand/1*, est statistiquement différente)

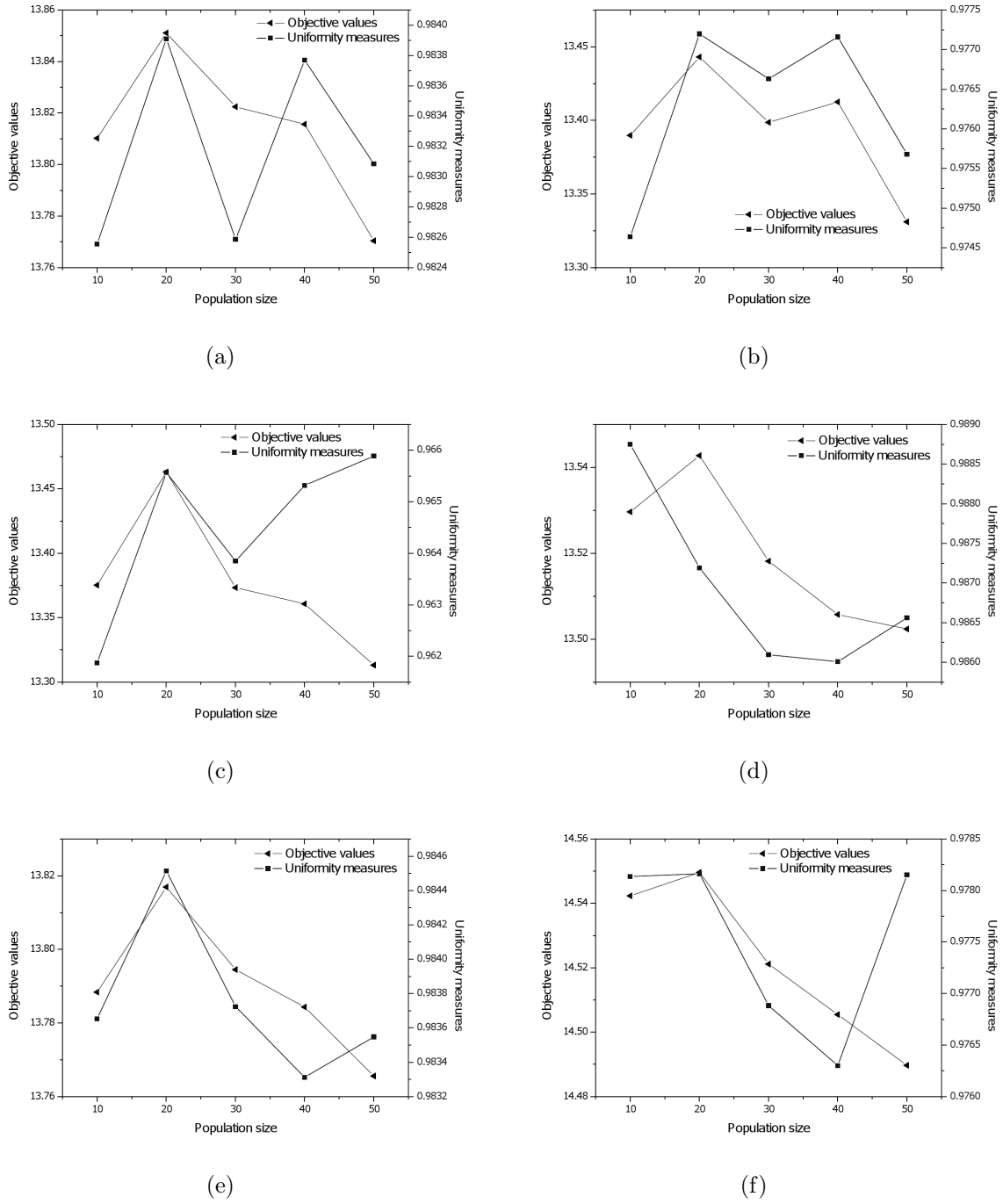
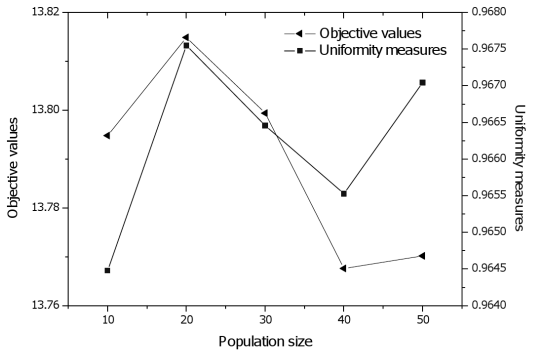
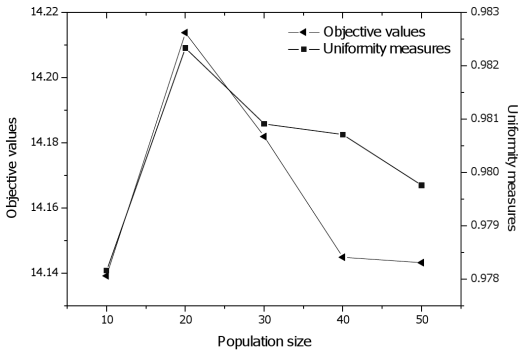


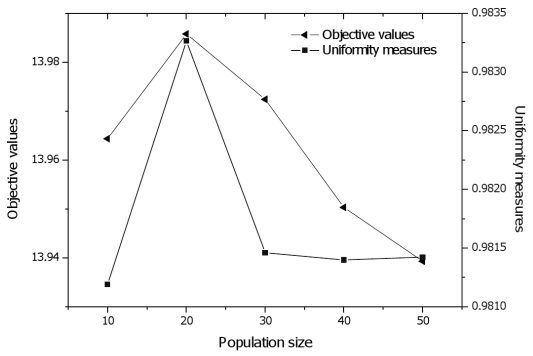
FIGURE 5.9: Effet de la variation de la taille de la population sur les performances de DBBO-Fuzzy : (a) Lena, (b) Peppers, (c) Cameraman, (d) Airplane, (e) Lake, (f) Walking bridge



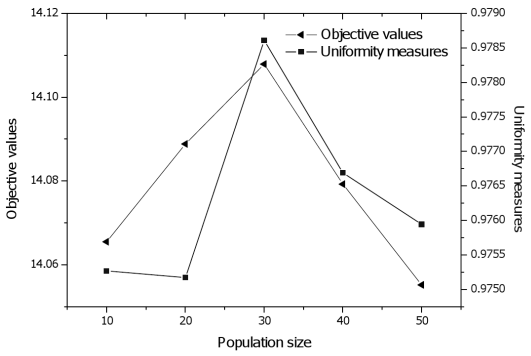
(g)



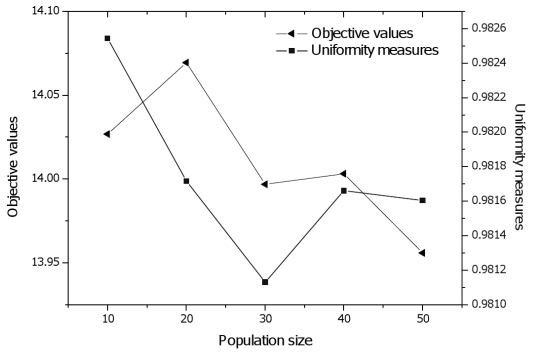
(h)



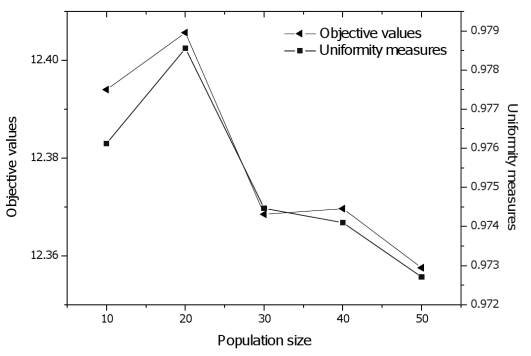
(i)



(j)



(k)



(l)

FIGURE 5.9: *(Cont.)*Effet de la variation de la taille de la population sur les performances de DBBO-Fuzzy : (g) Mandrill, (h) Barbara, (i) Boat, (j) Elaine, (k) GoldHill, (l) Fingerprint

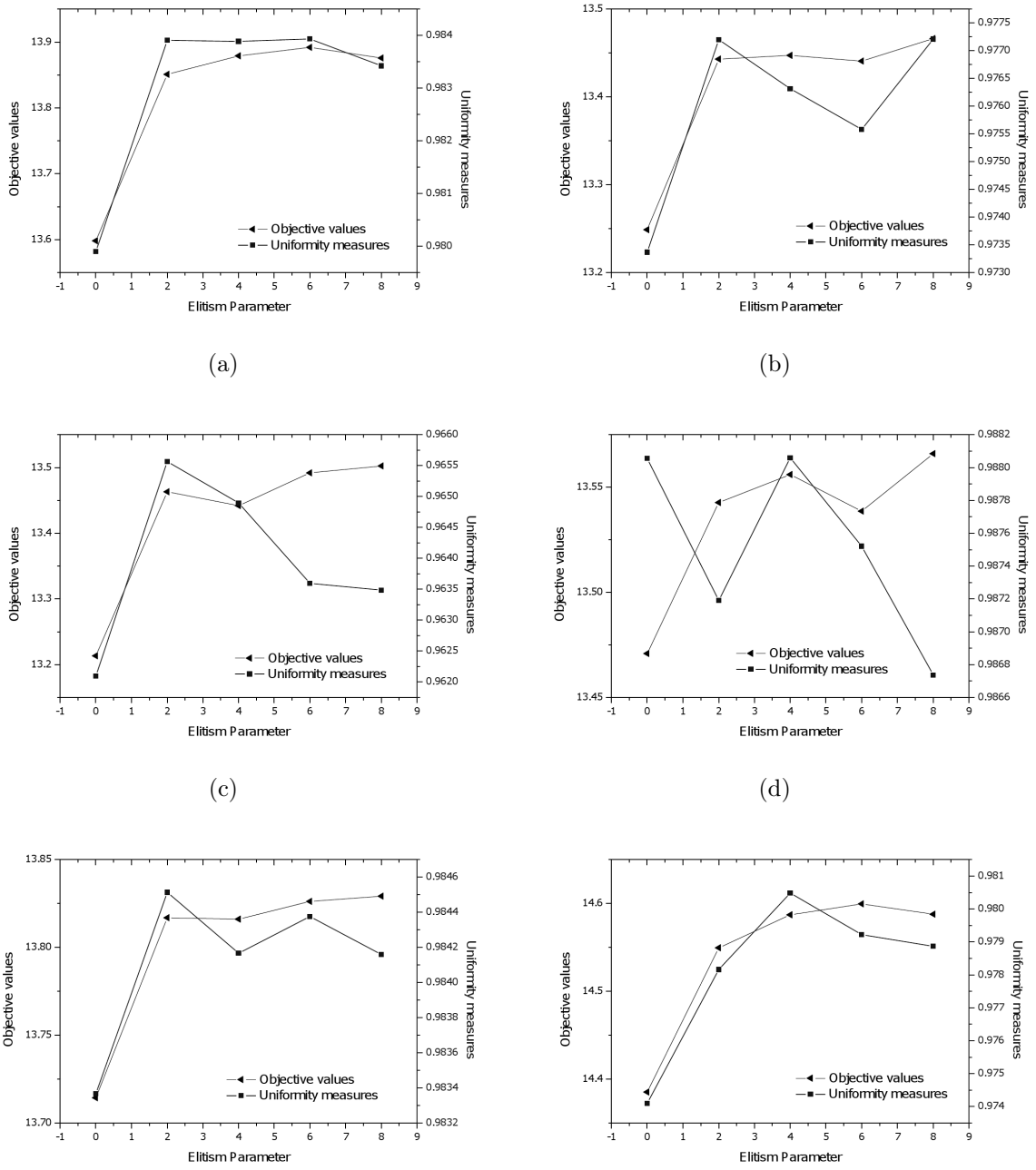
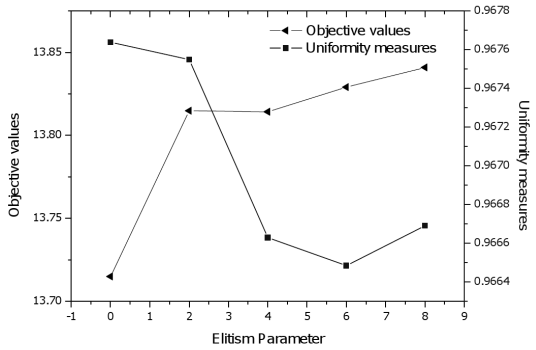
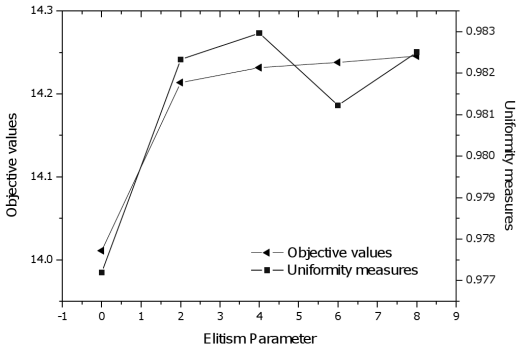


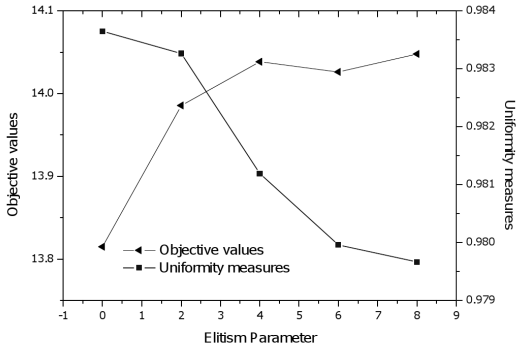
FIGURE 5.10: Effet de la variation du paramètre d'élitisme sur les performances de DBBO-Fuzzy : (a) Lena, (b) Peppers, (c) Cameraman, (d) Airplane, (e) Lake, (f) Walking bridge



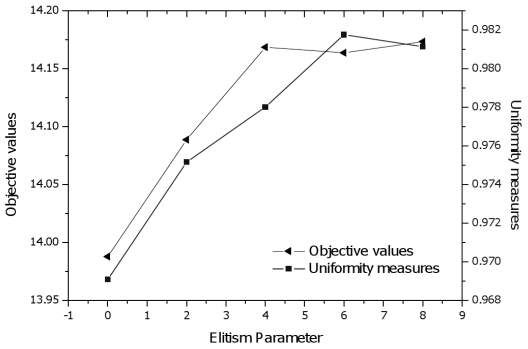
(g)



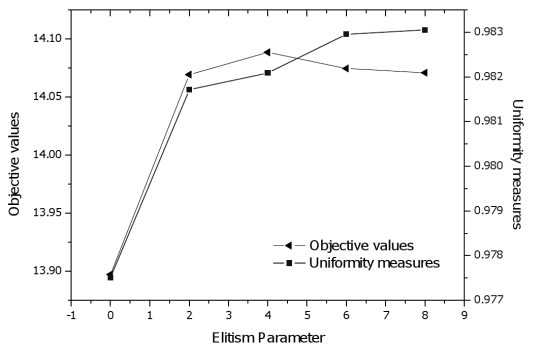
(h)



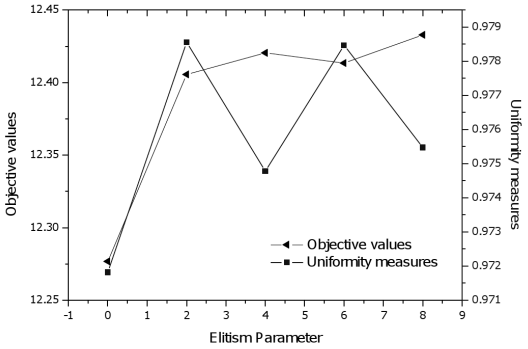
(i)



(j)



(k)



(l)

FIGURE 5.10: *(Cont.)*Effet de la variation du paramètre d'élitisme sur les performances de DBBO-Fuzzy : (g) Mandrill, (h) Barbara, (i) Boat, (j) Elaine, (k) GoldHill, (l) Fingerprint

Conclusion et perspectives

Cette thèse avait pour objet la résolution de problèmes d'optimisation globale à variables continues avec et sans contraintes. Nous avons présenté, dans un premier temps, un état de l'art sur les méthodes de la littérature permettant de résoudre les problèmes d'optimisation difficiles. Des contributions pour l'amélioration de l'algorithme BBO ont été proposées dans les domaines considérés. Nous avons mis en place une hybridation de BBO avec l'algorithme DE. Ce dernier a retenu notre attention pour ses performances, en terme de vitesse de convergence et de qualité de la solution. L'algorithme proposé dans le cadre de l'optimisation continue sans contraintes, nommé *2-StageBBO-DE*, obtient des résultats qui dépassent de manière très significative ceux obtenus par BBO et DE sur les principaux jeux de tests pour les problèmes d'optimisation difficile en variables continues. Ensuite, BBO a servi de base pour proposer d'autres algorithmes encore plus performants, pour résoudre des problèmes d'optimisation sous contraintes. L'élaboration de nouvelles méthodes, nommées CBBO, CBBO-DE et CBBO-PM, a nécessité l'intégration de mécanismes pour la prise en compte des contraintes. Nous avons validé notre approche sur des jeux de données classiques de la littérature en comparant nos résultats à ceux d'autres travaux.

En perspective, une analyse expérimentale plus poussée reste à faire. Les paramètres des algorithmes proposés peuvent également être *auto-adaptés*, afin de faciliter leur utilisation. De plus, les algorithmes proposés dans cette thèse sont mono-objectifs, c'est-à-dire qu'ils ne peuvent optimiser qu'une seule fonction objectif à la fois. Néanmoins, de nombreux problèmes réels d'optimisation nécessitent de pouvoir optimiser plusieurs fonctions objectifs simultanément. En perspective, il pourrait être intéressant de proposer une version *multiobjectif* de BBO. Une piste aussi intéressante concerne l'utilisation de différents modèles de migration et examiner leur influence sur les performances des algorithmes développés.

Après avoir validé les approches proposées sur les *benchmarks* de fonctions analytiques, nous nous sommes focalisés sur leur application à des problèmes réels. Le développement des techniques proposées a donné lieu à deux types d'applications, présentées dans les chapitres 4 et 5. La première application se trouve au carrefour de deux domaines : celui des métaheuristiques, et celui de la théorie de la détection, notamment à travers l'approche bayésienne. Les

métaheuristiques connues pour leur flexibilité et leur robustesse, ont été choisies comme méthode de résolution du problème d'allocation optimale de puissance dans les réseaux de capteur sans fil. L'objectif est d'optimiser la consommation énergétique des nœuds afin d'améliorer les performances du réseau et maximiser sa durée de vie. Dans ce travail, le centre de fusion est supposé avoir une connaissance préalable de l'état du canal, qui est sujet à un évanouissement de type *Rayleigh*. Quand il y a des changements dans le canal, l'allocation de puissance est mise à jour en conséquence. Cette tâche, d'ordinaire si facile à effectuer dans un réseau fiable avec des changements peu fréquents, devient rapidement problématique dans le cas d'un réseau sujet à la fois à des perturbations environnementales constantes et à des défaillances fréquentes. Dans le cas où les conditions du canal changent rapidement, l'allocation de puissance devrait être mise à jour de façon dynamique afin d'assurer une performance optimale. Cependant, il existe d'autres alternatives potentielles en utilisant par exemple des méthodes d'*optimisation dynamique*.

Nous avons également abordé le problème de segmentation d'images par seuillage. Ce travail se situe résolument à la croisée de trois domaines, celui de l'optimisation, de l'analyse d'images et celui de la théorie des ensembles flous. Une approche hybride, DBBO-Fuzzy, a été utilisée pour la résolution du problème de seuillage à trois niveaux d'images en niveau de gris. Le critère de segmentation que nous avons proposé d'utiliser repose sur la maximisation de l'entropie floue. DBBO-Fuzzy a été testé sur différentes images et nous avons pu observer sa robustesse en terme de qualité de solutions trouvées et son efficacité pour déterminer les paramètre optimaux qui maximisent l'entropie. Des travaux futurs pourrait s'intéresser à l'emploi d'une approche multicritère. Au lieu de considérer une fonction objectif unique, un modèle biobjectif pourrait être adopté pour le problème de seuillage à trois niveaux, dans lequel on chercherait à optimiser simultanément l'entropie totale floue et le facteur d'uniformité.

Annexe - Les fonctions tests pour l'optimisation sous contraintes

Dans cet annexe, nous présentons la description des 12 fonctions de test, utilisées dans la partie expérimentale de l'optimisation sous contraintes.

A. g01

Minimiser [Floudas & Pardalos, 1990] :

$$f(x) = 5\sum_{i=1}^{4} x_i - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i$$

Sous les contraintes suivantes :

$$\begin{aligned}
g_1(x) &= 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \leq 0 \\
g_2(x) &= 2x_1 + 2x_3 + x_{10} + x_{12} - 10 \leq 0 \\
g_3(x) &= 2x_2 + 2x_3 + x_{11} + x_{12} - 10 \leq 0 \\
g_4(x) &= -8x_1 + x_{10} \leq 0 \\
g_5(x) &= -8x_2 + x_{11} \leq 0 \\
g_6(x) &= -8x_3 + x_{12} \leq 0 \\
g_7(x) &= -2x_4 - x_5 + x_{10} \leq 0 \\
g_8(x) &= -2x_6 - x_7 + x_{11} \leq 0 \\
g_9(x) &= -2x_8 - x_9 + x_{12} \leq 0
\end{aligned}$$

où $0 \leq x_i \leq 1$ pour tout $i = 1, \ldots, 9$, $0 \leq x_i \leq 100$ (i = 10, 11, 12) et $0 \leq x_{13} \leq 1$. Le minimum global est $x^* = (1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 1)$ dont la valeur est $f(x^*) = -15$. Seulement six des neufs contraintes sont actives au niveau de l'optimum global (g_1, g_2, g_3, g_7, g_8 and g_9).

B. g02

Maximiser [Koziel & Michalewicz, 1999] :

$$f(x) = \left| \frac{\sum_{i=1}^{n} cos^4(x_i) - 2\prod_{i=1}^{n} cos^2(x_i)}{\sqrt{\sum_{i=1}^{n} i x_i^2}} \right|$$

Sous les contraintes suivantes :

$$g_1(x) = 0.75 - \prod_{i=1}^{n} x_i \leq 0$$

$$g_2(x) = \sum_{i=1}^{n} x_i - 7.5n \leq 0$$

où $n = 20$ et $0 < x_i \leq 10$ pour tout $i = 1, \ldots, n$. Le maximum global est inconnu ; la meilleure solution trouvée est $f(x^*) = 0.803619$, pour lequel la contrainte g_1 est presque active ($g_1 = -10^{-8}$).

C. g03

Maximiser [Michalewicz et al., 1996] :

$$f(x) = \left(\sqrt{n}\right)^n \prod_{i=1}^{n} x_i$$

Sous la contrainte suivante :

$$h_1(x) = \sum_{i=1}^{n} x_i^2 - 1 = 0$$

où $n = 10$ et $0 \leq x_i \leq 1$ pour tout $i = 1, \ldots, n$. L'optimum global est $x^* = 1/\sqrt{n}$ (i = 1, ..., n) où $f(x^*) = 1$.

D. g04

Minimiser [Himmelblau, 1972] :

$$f(x) = 5.3578547x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141$$

Sous les contraintes suivantes :

$$\begin{aligned}
g_1(x) &= 85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4 - 0.0022053x_3x_5 - 92 \leq 0 \\
g_2(x) &= -85.334407 - 0.0056858x_2x_5 - 0.0006262x_1x_4 + 0.0022053x_3x_5 \leq 0 \\
g_3(x) &= 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^2 - 110 \leq 0 \\
g_4(x) &= -80.51249 - 0.0071317x_2x_5 - 0.0029955x_1x_2 - 0.0021813x_3^2 + 90 \leq 0 \\
g_5(x) &= 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4 - 25 \leq 0 \\
g_6(x) &= -9.300961 - 0.0047026x_3x_5 - 0.0012547x_1x_3 - 0.0019085x_3x_4 + 20 \leq 0
\end{aligned}$$

où $78 \leq x_1 \leq 102$, $33 \leq x_2 \leq 45$ and $27 \leq x_i \leq 45$ (i = 3,4,5). La solution optimale est x^* = (78, 33, 29.995256025682, 45, 36.775812905788) où $f(x^*) = -30665.539$. Deux contraintes sont actives au niveau de l'optimum (g_1 et g_6).

E. g05

Minimiser [Hock & Schittkowski, 1981] :

$$f(x) = 3x_1 + 0.000001x_1^3 + 2x_2 + (0.000002/3)x_2^3$$

Sous les contraintes suivantes :

$$\begin{aligned}
g_1(x) &= -x_4 + x_3 - 0.55 \leq 0 \\
g_2(x) &= -x3 + x4 - 0.55 \leq 0 \\
h_3(x) &= 1000sin(-x_3 - 0.25) + 1000sin(-x_4 - 0.25) + 894.8 - x1 = 0 \\
h_4(x) &= 1000sin(x_3 - 0.25) + 1000sin(x_3 - x_4 - 0.25) + 894.8 - x_2 = 0 \\
h_5(x) &= 1000sin(x_4 - 0.25) + 1000sin(x_4 - x_3 - 0.25) + 1294.8 = 0
\end{aligned}$$

où $0 \leq x_1 \leq 1200$, $0 \leq x_2 \leq 1200$, $-0.55 \leq x_3 \leq 0.55$ et $-0.55 \leq x_4 \leq 0.55$. La meilleure solution connue pour ce problème est $x^* = (679.9453, 1026.067, 0.1188764, -0.3962336)$ avec $f(x^*) = 5126.4981$ [Koziel & Michalewicz, 1999].

F. g06

Minimiser [Floudas & Pardalos, 1990] :

$$f(x) = (x_1 - 10)^3 + (x_2 - 20)^3$$

Sous les contraintes suivantes :

$$g_1(x) = -(x_1 - 5)^2 - (x_2 - 5)^2 + 100 \leq 0$$
$$g_2(x) = (x1 - 6)^2 + (x_2 - 5)^2 - 82.81 \leq 0$$

où $13 \leq x_1 \leq 100$ et $0 \leq x_2 \leq 100$. La solution optimale est $x^* = (14.095, 0.84296)$ avec $f(x^*) = -6961.81388$. Les deux contraintes du problèmes sont actives.

G. g07

Minimiser [Hock & Schittkowski, 1981] :

$$\begin{aligned} f(x) = \quad & x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 + 4(x_4 - 5)^2 + (x_5 - 3)^2 \\ & +2(x_6 - 1)^2 + 5x_7^2 + 7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_10 - 7)^2 + 45 \end{aligned}$$

Sous les contraintes suivantes :

$$\begin{aligned} g_1(x) &= -105 + 4x_1 + 5x_2 - 3x_7 + 9x_8 \leq 0 \\ g_2(x) &= 10x_1 - 8x_2 - 17x_7 + 2x_8 \leq 0 \\ g_3(x) &= -8x_1 + 2x_2 + 5x_9 - 2x_10 - 12 \leq 0 \\ g_4(x) &= 3(x_1 - 2)^2 + 4(x_2 - 3)^2 + 2x_3^2 - 7x_4 - 120 \leq 0 \\ g_5(x) &= 5x_1^2 + 8x_2 + (x_3 - 6)^2 - 2x_4 - 40 \leq 0 \\ g_6(x) &= x_1^2 + 2(x_2 - 2)^2 - 2x_1x_2 + 14x_5 - 6x_6 \leq 0 \\ g_7(x) &= 0.5(x_1 - 8)^2 + 2(x_2 - 4)^2 + 3x_5^2 - x_6 - 30 \leq 0 \\ g_8(x) &= -3x_1 + 6x_2 + 12(x_9 - 8)^2 - 7x_{10} \leq 0 \end{aligned}$$

où $-10 \leq x_i \leq 10$ (i = 1, ...,10). Ce problème a 3 contraintes linéaires et 5 autres non linéaires. L'optimum global est x^* = (2.171996, 2.363683, 8.773926, 5.095984, 0.9906548, 1.430574, 1.321644, 9.828726, 8.280092, 8.375927) avec $f(x^*) = 24.3062091$. Six contraintes sont actives aux alentours de l'optimum global (g_1, g_2, g_3, g_4, g_5 et g_6).

H. g08

Minimiser [Koziel & Michalewicz, 1999] :

$$f(x) = -\frac{sin^3(2\pi x_1)sin(2\pi x_2)}{x_1^3(x_1 + x_2)}$$

Sous les contraintes suivantes :

$$g_1(x) = -x_1^2 - x_2 + 1 \leq 0$$
$$g_2(x) = 1 - x_1 + (x_2 - 4)^2 \leq 0$$

où $0 \leq x_1 \leq 10$ et $0 \leq x_2 \leq 10$. La fonction a plusieurs optima locaux, les plus hauts pics se trouvent le long de l'axe des x. L'optimum connu se trouve à l'intérieur de la région faisable et a comme coordonnées x^* = (1.2279713, 4.2453733) avec $f(x^*) = 0.095825$.

I. g09

Minimiser [Hock & Schittkowski, 1981] :

$$f(x) = (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 + 10x_5^6 + 7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7$$

Sous les contraintes suivantes :

$$\begin{aligned} g_1(x) &= -127 + 2x_1^2 + 3x_2^4 + x_3 + 4x_4^2 + 5x_5 \leq 0 \\ g_2(x) &= -282 + 7x_1 + 3x_2 + 10x_3^2 + x_4 - x_5 \leq 0 \\ g_3(x) &= -196 + 23x_1 + x_2^2 + 6x_6^2 - 8x_7 \leq 0 \\ g_4(x) &= 4x_1^2 + x_2^2 - 3x_1x_2 + 2x_3^2 + 5x_6 - 11x_7 \leq 0 \end{aligned}$$

où $-10 \leq x_i \leq 10$ pour (i = 1, ..., 7). La solution optimale est x^* = (2.330499, 1.951372, −0.4775414, 4.365726, −0.6244870, 1.038131, 1.594227) avec $f(x^*) = 680.6300573$. Deux contraintes sont actives au niveau de l'optimum global ($g1$ et $g4$).

J. g10

Minimiser [Hock & Schittkowski, 1981] :

$$f(x) = x_1 + x_2 + x_3$$

Sous les contraintes suivantes :

$$\begin{aligned} g_1(x) &= -1 + 0.0025(x_4 + x_6) \leq 0 \\ g_2(x) &= -1 + 0.0025(x_5 + x_7 - x_4) \leq 0 \\ g_3(x) &= -1 + 0.01(x_8 - x_5) \leq 0 \\ g_4(x) &= -x_1x_6 + 833.33252x_4 + 100x_1 - 83333.333 \leq 0 \\ g_5(x) &= -x_2x_7 + 1250x_5 + x_2x_4 - 1250x_4 \leq 0 \\ g_6(x) &= -x_3x_8 + 1250000 + x_3x_5 - 2500x_5 \leq 0 \end{aligned}$$

où $100 \leq x_1 \leq 10000$, $1000 \leq x_i \leq 10000$ (i = 2, 3) et $10 \leq x_i \leq 1000$ (i = 4, ..., 8). Le minimum global est x^* = (579.3167, 1359.943, 5110.071, 182.0174, 295.5985, 217.9799, 286.4162, 395.65979), avec $f(x^*) = 7049.3307$. Toutes les contraintes sont actives au niveau de l'optimum global ($g1$, $g2$ et $g3$).

K. g11

Minimiser [Koziel & Michalewicz, 1999] :

$$f(x) = x_1^2 + (x_2 - 1)^2$$

Sous la contrainte suivante :

$$h(x) = x_2 - x_1^2 = 0$$

où $-1 \leq x_1 \leq 1$ et $-1 \leq x_2 \leq 1$. L'optimum global est $x^* = (\pm 1/\sqrt{2}, 1/2)$ avec $f(x^*) = 0.75$.

L. g12

Maximiser [Koziel & Michalewicz, 1999] :

$$f(x) = \big(100 - (x_1 - 5)^2 - (x_2 - 5)^2 - (x_3 - 5)^2\big)/100$$

Sous la contrainte suivante :

$$g(x) = (x_1 - p)^2 + (x_2 - q)^2 + (x_3 - r)^2 - 0.0625 \leq 0$$

où $0 \leq x_i \leq 10$ (i=1, 2, 3) et $p, q, r = 1, 2, \ldots, 9$. La région faisable de l'espace de recherche se compose de 9^3 sphères disjointes. Un point (x_1, x_2, x_3) est faisable si et seulement s'il existe p, q, r tel que l'inégalité ci-dessus est satisfaite. L'optimum se situe au point x^* = (5,5,5) avec $f(x^*) = 1$. La solution se situe dans la région réalisable.

Références bibliographiques

[Abadir & Magnus, 2005] K.M. Abadir & J.R. Magnus. *Matrix algebra. Econometric exercises 1.* Cambridge, United Kingdom : Cambridge University Press, 2005.

[Aikelin & Cayzer, 2002] U. Aikelin & S. Cayzer. The Danger Theory and Its Application to Artificial Immune Systems. In *Proceedings of the 1st International Conference on Artificial Immune Systems*, pp. 141–148, 2002.

[Akyildiz et al., 2002] I. F. Akyildiz, W. Su, Y. Sankasubramaniam, & E. Cayirci. Wireless sensor networks : A survey. *Computer Networks*, 38: 393–422, 2002.

[Amirjanov, 2006] A. Amirjanov. The development of a changing range genetic algorithm. In *Computer Methods in Applied Mechanics and Engineering*, pp. 2495–2508, 2006.

[Bäck, 1996] T. Bäck. *Evolutionary algorithms in theory and practice - evolution strategies, evolutionary programming, genetic algorithms.* Oxford University Press, 1996. ISBN 978-0-19-509971-3.

[Bäck et al., 1991] T. Bäck, F. Hoffmeister, & H.-P. Schwefel. A survey of evolution strategies. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pp. 2–9, 1991.

[Bäck et al., 1993] T. Bäck, G. Rudolph, & H.-P. Schwefel. Evolutionary programming and evolution strategies : Similarities and differences. In *Proceedings of the Second Annual Conference on Evolutionary Programming*, pp. 11–22, 1993.

[Battiti & Tecchiolli, 1994] R. Battiti & G. Tecchiolli. The reactive tabu search. *ORSA Journal on Computing*, 6(2): 126–140, 1994.

[Becerra & Coello, 2005] R. L. Becerra & C. A. C. Coello. Optimization with constraints using a cultured differential evolution approach. In *Proceedings of the 2005 conference on Genetic and evolutionary computation*, GECCO '05, pp. 27–34, New York, NY, USA, 2005. ACM. ISBN 1-59593-010-8.

[Bhandari & Pal, 1993] D. Bhandari & N. R. Pal. Some new information measures for fuzzy sets. *Inf. Sci.*, 67: 209–228, January 1993. ISSN 0020-0255.

[Blickle & Thiele, 1995] T. Blickle & L. Thiele. A comparison of selection schemes used in genetic algorithms. *Evolutionary Computation*, 4(11): 311–347, 1995.

[Bloch et al., 2003] I. Bloch, J.-P. Le Cadre, & H. Maître. Approches probabilistes et statistiques. In I. Bloch, editor, *Fusion d'informations en traitement du signal et des images*, Traité IC2, chapter 6, pp. 87–118. Hermès, 2003.

[Bonabeau et al., 1999] E. Bonabeau, M. Dorigo, & G. Theraulaz. *Swarm intelligence : from natural to artificial systems*. Oxford University Press, Inc., New York, NY, USA, 1999. ISBN 0195131592.

[Bouchon-Meunier & Marsala, 2003] B. Bouchon-Meunier & C. Marsala. *Logique floue, principes, aide à la décision*. Hermès Science, 2003.

[Boussaïd et al., 2011a] I. Boussaïd, A. Chatterjee, P. Siarry, & M. Ahmed-Nacer. Two-stage update biogeography-based optimization using differential evolution algorithm (dbbo). *Computers & Operations Research*, 38: 1188–1198, August 2011. ISSN 0305-0548.

[Boussaïd et al., 2011b] I. Boussaïd, A. Chatterjee, P. Siarry, & M. Ahmed-Nacer. Hybridizing biogeography-based optimization with differential evolution for optimal power allocation in wireless sensor networks. *IEEE Transactions on Vehicular Technology*, 60(5): 2347–2353, June 2011.

[Boussaïd et al., 2012] I. Boussaïd, A. Chatterjee, P. Siarry, & M. Ahmed-Nacer. Biogeography-based optimization for constrained optimization problems. *Computers & Operations Research*, 39: 3293–3304, December 2012.

[Boussaïd et al., 2013a] I. Boussaïd, A. Chatterjee, P. Siarry, & M. Ahmed-Nacer. Hybrid bbo-de algorithms for fuzzy entropy-based thresholding. In Amitava Chatterjee & Patrick Siarry, editors, *Computational Intelligence in Image Processing*, pp. 37–69. Springer Berlin Heidelberg, 2013. ISBN 978-3-642-30620-4. doi : 10.1007/978-3-642-30621-1_3.

[Boussaïd et al., 2013b] I. Boussaïd, A. Chatterjee, P. Siarry, & M. Ahmed-Nacer. A comparative study of modified bbo variants and other metaheuristic optimization techniques for the optimal power allocation in wireless sensor networks. In Amitava Chatterjee, Hadi Nobahari, & Patrick Siarry, editors, *Advances in Heuristic Signal Processing and Applications*, pp. 79–110. Springer Berlin Heidelberg, 2013. ISBN 978-3-642-37879-9.

[Boussaïd et al., 2013c] I. Boussaïd, J. Lepagnot, & P. Siarry. A survey on optimization metaheuristics. *Information Sciences*, 237: 8–117, 2013. ISSN 0020-0255. doi : 10.1016/j.ins.2013.02.041.

[Braviano, 1995] G. Braviano. *Logique floue en segmentation d'images : seuillage par entropie et structures pyramidales irrégulières*. PhD thesis, Université Joseph Fourier - Grenoble, France, 1995.

[Cerny, 1985] V. Cerny. Thermodynamical approach to the traveling salesman problem : an efficient simulation algorithm. *Journal of Optimization Theory and Applications*, 45(1): 41–51, 1985.

[Chamberland & Veeravalli, 2004] J. F. Chamberland & V. V. Veeravalli. Decentralized detection in wireless sensor systems with dependent observations. In *International Conference on Computing, Communications and Control Technologies*, Austin, TX, Aug. 2004.

[Chang et al., 2006] C. I. Chang, Y. Du, J. Wang, S. M. Guo, & P. D. Thouin. Survey and comparative analysis of entropy and relative entropy thresholding techniques. *Vision, Image and Signal Processing, IEE Proceedings* -, 153(6): 837–850, 2006.

[Charnes & Cooper, 1967] A. Charnes & W. W. Cooper. *Management models and industrial applications of linear programming*. Wiley, New York [u.a.], 1967.

[Cheng et al., 1998] H.D. Cheng, J.R. Chen, & J.G. Li. Threshold selection based on fuzzy c-partition entropy approach. *Pattern Recognition*, 31(7): 857–870, July 1998.

[Clerc, 2003] M. Clerc. Tribes - un exemple d'optimisation par essaim particulaire sans paramètres de contrôle. In *Optimisation par Essaim Particulaire (OEP 2003)*, 2003.

[Clerc & Kennedy, 2002] M. Clerc & J. Kennedy. The particle swarm - explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, 6(1): 58–73, 2002.

[Coello & Becerra, 2002] C. A. C. Coello & R. L. Becerra. Adding knowledge and efficient data structures to evolutionary programming : A cultural algorithm for constrained optimization. In *GECCO*, pp. 201–209, 2002.

[Coello et al., 1995] C. A. C. Coello, A. D. Christiansen, & A. H. Aguirre. Multiobjective design optimization of counterweight balancing of a robot arm using genetic algorithms.

In *Proceedings of the Seventh International Conference on Tools with Artificial Intelligence*, TAI '95, pp. 20–, Washington, DC, USA, 1995. IEEE Computer Society. ISBN 0-8186-7312-5.

[Cook, 1971] S. A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, STOC '71, pp. 151–158, New York, NY, USA, 1971. ACM. doi : 10.1145/800157.805047.

[Cover & Thomas, 1991] T. M. Cover & J. A. Thomas. *Elements of information theory*. Wiley-Interscience, New York, NY, USA, 1991. ISBN 0-471-06259-6.

[Darwin, 1859] C. Darwin. *On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life*. J. Murray, June 1859. ISBN 0486450066.

[Dasgupta et al., 2011] D. Dasgupta, S. Yu, & F. Nino. Recent advances in artificial immune systems : Models and applications. *Applied Soft Computing*, 11(2): 1574–1587, 2011.

[de Castro & Von Zuben, 2002] L. N. de Castro & F. J. Von Zuben. Learning and optimization using the clonal selection principle. *IEEE Trans. Evolutionary Computation*, 6(3): 239–251, 2002.

[De Luca & Termini, 1972] A. De Luca & S. Termini. A definition of a non-probabilistic entropy in the setting of fuzzy sets theory. *Information and Control*, 20: 301–312, 1972.

[Deb, 2000] K. Deb. An efficient constraint handling method for genetic algorithm. *Computer Methods in Applied Mechanics and Engineering*, 186: 311–338, October 2000.

[Deb & Agrawal, 1995] K. Deb & R.B. Agrawal. Simulated binary crossover for continuous search space. *Complex System*, 9: 115–148, 1995.

[Deb et al., 2000] K. Deb, S. Agrawal, A. Pratap, & T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimisation : Nsga-ii. In *Proceedings of the 6th International Conference on Parallel Problem Solving from Nature*, PPSN VI, pp. 849–858, London, UK, UK, 2000. Springer-Verlag. ISBN 3-540-41056-2.

[DeJong, 1975] K.A. DeJong. *An analysis of the behavior of a class of genetic adaptive systems*. PhD thesis, University of Michigan, Ann Arbor, MI, USA, 1975.

[Deneubourg et al., 1990] J. L. Deneubourg, S. Aron, S. Goss, & J. M. Pasteels. The self-organizing exploratory pattern of the argentine ant. *Journal of Insect Behavior*, 3(2): 159–168, March 1990. ISSN 0892-7553.

[Dorigo, 1992] M. Dorigo. *Optimization, Learning and Natural Algorithms*. PhD thesis, Politecnico di Milano, Italy, 1992.

[Dorigo & Blum, 2005] M. Dorigo & C. Blum. Ant colony optimization theory : A survey. *Theoretical Computer Science*, 344(2-3): 243–278, November 2005. ISSN 03043975. doi : 10.1016/j.tcs.2005.05.020.

[Dorigo et al., 1991] M. Dorigo, V. Maniezzo, & A. Colorni. Positive feedback as a search strategy. Technical Report 91-016, Dipartimento di Elettronica, Politecnico di Milano, Milan, Italy, 1991.

[Dorigo et al., 1996] M. Dorigo, V. Maniezzo, & A. Colorni. The ant system : Optimization by a colony of cooperating agents. *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS-PART B*, 26(1): 29–41, 1996.

[Dorigo & Thomas, 2010] Marco Dorigo & Stützle Thomas. Ant colony optimization : Overview and recent advances. In Michel Gendreau & Jean-Yves Potvin, editors, *Handbook of Metaheuristics*, International Series in Operations Research & Management Science, pp. 227–263. Springer US, 2010. ISBN 978-1-4419-1665-5.

[Dow, 2003] M. Dow. Explicit inverses of toeplitz and associated matrices. *ANZIAM J.*, 44 (E): E185–E215, January 2003.

[Dréo et al., 2003] J. Dréo, A. Petrowski, É. D. Taillard, & P. Siarry. *Métaheuristiques pour l'optimisation difficile*. Eyrolles (Editions), November 2003. ISBN 2212113684.

[Du et al., 2009] D. Du, D. Simon, & M. Ergezer. Biogeography-based optimization combined with evolutionary strategy and immigration refusal. In *SMC'09 : Proceedings of the 2009 IEEE international conference on Systems, Man and Cybernetics*, pp. 997–1002, Piscataway, NJ, USA, 2009. IEEE Press. ISBN 978-1-4244-2793-2.

[Dubois & Prade, 1988] D. Dubois & H. Prade. *Théorie des possibilités : application à la représentation des connaissances en informatique*. Masson, Paris, 1988.

[Dueck & Scheuer, 1990] G. Dueck & T. Scheuer. Threshold accepting : A general purpose optimization algorithm appearing superior to simulated annealing. *Journal of Computational Physics*, 90(1): 161–175, 1990.

[Eberhart et al., 1996] R. Eberhart, P. Simpson, & R. Dobbins. *Computational intelligence PC tools*. Academic Press Professional, Inc., San Diego, CA, USA, 1996. ISBN 0-12-228630-8.

[Edgeworth, 1885] F. Y. Edgeworth. Methods of statistics. *Journal of the Statistical Society of London, Jubilee Volume*, pp. 181 – 217, 1885.

[Ergezer et al., 2009] M. Ergezer, D. Simon, & D. Du. Oppositional biogeography-based optimization. In *SMC'09 : Proceedings of the 2009 IEEE international conference on Systems, Man and Cybernetics*, pp. 1009–1014, Piscataway, NJ, USA, 2009. IEEE Press. ISBN 978-1-4244-2793-2.

[Fan & Xie, 1999] J. Fan & W Xie. Distance measure and induced fuzzy entropy. *Fuzzy Sets Syst.*, 104: 305–314, June 1999. ISSN 0165-0114.

[Farmer et al., 1986] J. D. Farmer, N. H. Packard, & A. S. Perelson. The immune system, adaptation, and machine learning. *Phys. D*, 2(1-3): 187–204, 1986. ISSN 0167-2789.

[Feo & Resende, 1989] T. A. Feo & M. G. C. Resende. A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, 8(2): 67–71, 1989.

[Feo & Resende, 1995] T. A. Feo & M. G. C. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6(2): 109–133, 1995.

[Festa & Resende, 2009a] P. Festa & M. Resende. An annotated bibliography of GRASP, Part I : Algorithms. *International Transactions in Operational Research*, 16(1): 1–24, 2009.

[Festa & Resende, 2009b] P. Festa & M. Resende. An annotated bibliography of GRASP, Part II : Applications. *International Transactions in Operational Research*, 16(2): 131–172, 2009.

[Floudas & Pardalos, 1990] C. A. Floudas & P. M. Pardalos. *A collection of test problems for constrained global optimization algorithms*. Springer-Verlag New York, Inc., New York, NY, USA, 1990.

[Fogel, 1991] D. B. Fogel. *System Identification through Simulated Evolution : A Machine Learning Approach to Modeling*. Ginn Press, 1991. ISBN 0536579431.

[Fogel, 1995] D. B. Fogel. *Evolutionary computation : toward a new philosophy of machine intelligence*. IEEE Press, Piscataway, NJ, USA, 1995. ISBN 0-7803-1038-1.

[Fogel et al., 1966] L. J. Fogel, A. J. Owens, & M. J. Walsh. *Artificial Intelligence through Simulated Evolution*. John Wiley, New York, USA, 1966.

[Fonseca & Fleming, 1993] C. M. Fonseca & P. J. Fleming. Genetic algorithms for multiobjective optimization : Formulation , discussion and generalization. *Proceedings of the Fifth International Conference on Genetic Algorithms (ICGA'93)*, 1: 416–423, June 1993.

[Forrest et al., 1994] Stephanie Forrest, A. S. Perelson, L. Allen, & R. Cherukuri. Self-nonself discrimination in a computer. In *Proceedings of the Symposium on Research in Security and Privacy*, pp. 202–212, 1994.

[Freixenet et al., 2002] J. Freixenet, X. Muñoz, D. Raba, J. Martí, & X. Cufí. Yet another survey on image segmentation : Region and boundary information integration. In *Proceedings of the 7th European Conference on Computer Vision-Part III*, ECCV '02, pp. 408–422, London, UK, UK, 2002. Springer-Verlag. ISBN 3-540-43746-0.

[Gardeux, 2011] V. Gardeux. *Conception d'heuristiques d'optimisation pour les problèmes de grande dimension. Application à l'analyse de données de puces à ADN.* PhD thesis, Paris-Est University, Créteil, France, November 30, 2011.

[Glover, 1986] F. Glover. Future paths for integer programming and links to artificial intelligence. *Computers and Operations Research*, 13(5): 533–549, 1986.

[Goldberg, 1989] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning.* Studies in Computational Intelligence. Addison-Wesley Longman Publishing Co., Inc., 1st edition, 1989. ISBN 0201157675.

[Goldberg & Deb, 1991] D. E. Goldberg & K. Deb. A comparative analysis of selection schemes used in genetic algorithms. In *Foundations of Genetic Algorithms*, pp. 69–93. Morgan Kaufmann, 1991.

[Gong et al., 2010a] W. Gong, Z. Cai, & C. Ling. DE/BBO : a hybrid differential evolution with biogeography-based optimization for global numerical optimization. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, March 2010. ISSN 1432-7643.

[Gong et al., 2010b] W. Gong, Z. Cai, C. Ling, & H. Li. A real-coded biogeography-based optimization with mutation. *Applied Mathematics and Computation*, 216(9): 2749–2758, 2010.

[Gonzalez & Woods, 1992] R. C. Gonzalez & R. E. Woods. *Digital image processing.* Addison-Wesley, 1992. ISBN 978-0-201-50803-1.

[Greensmith et al., 2005] Julie Greensmith, Uwe Aickelin, & Steve Cayzer. Introducing Dendritic Cells as a Novel Immune-Inspired Algorithm for Anomaly Detection. In *Artificial Immune Systems*, LNCS, pp. 153–167. Springer, 2005.

[Griewank, 1981] A.O. Griewank. Generalized descent for global optimization. *Journal of Optimization Theory and Applications*, 34(1): 11–39, 1981.

[Guiasu & Theodorescu, 1971] S. Guiasu & R. Theodorescu. Incertitude et information, 1971.

[Hadj-Alouane & Bean, 1992] A. B. Hadj-Alouane & J. C. Bean. A genetic algorithm for the multiple-choice integer program. Technical report tr 92-50, Department of Industrial and Operations Engineering, The University of Michigan, 1992.

[Hansen et al., 1995] N. Hansen, A. Ostermeier, & A. Gawelczyk. On the adaptation of arbitrary normal mutation distributions in evolution strategies : The generating set adaptation. In *Proceedings of the 6th International Conference on Genetic Algorithms*, pp. 57–64, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc. ISBN 1-55860-370-0.

[Hart & Shogan, 1987] J. P. Hart & A. W. Shogan. Semi-greedy heuristics : An empirical study. *Operations Research Letters*, 6(3): 107–114, 1987.

[Heppner & Grenander, 1990] F. Heppner & U. Grenander. A stochastic nonlinear model for coordinated bird flocks. In E. Krasner, editor, *The ubiquity of chaos*, pp. 233–238. AAAS Publications, 1990.

[Hillis, 1990] W. D. Hillis. Co-evolving parasites improve simulated evolution as an optimization procedure. *Phys. D*, 42: 228–234, June 1990. ISSN 0167-2789.

[Himmelblau, 1972] D. Himmelblau. *Applied Nonlinear Programming*. McGraw-Hill, New-York, USA, 1972.

[Hock & Schittkowski, 1981] W. Hock & K. Schittkowski. *Test Examples for Nonlinear Programming Codes*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1981. ISBN 0387105611.

[Holland, 1975] J. H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI, USA, 1975.

[Homaifar et al., 1994] A. Homaifar, C. X. Qi, & S. H. Lai. Constrained Optimization Via Genetic Algorithms. *SIMULATION*, 62(4): 242–253, April 1994. doi : 10.1177/003754979406200405.

[Horn et al., 1994] J. Horn, N. Nafpliotis, & D. E. Goldberg. A niched pareto genetic algorithm for multiobjective optimization. In *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*, pp. 82–87 vol.1, 1994.

[Jerne, 1973] N. K. Jerne. Towards a network theory of the immune system. *Annals of Immunology*, 125C(1-2): 373–389, January 1973. ISSN 0300-4910.

[Joines & Houck, 1994] J. A. Joines & C. R. Houck. On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GAs. In *Proceeding of the 1st IEEE Conference on Evolutionary Computation*, pp. 579–584. IEEE Press, 1994.

[Kapur, 1997] J. N. Kapur. *Measures of Fuzzy Information*. Mathematical Sciences Trust Society, New Delhi, 1997.

[Kapur et al., 1985] J. N. Kapur, P. K. Sahoo, & A. K. C. Wong. A new method for gray-level picture thresholding using the entropy of the histogram. *Computer Vision, Graphics, and Image Processing*, 29(3): 273–285, mars 1985. ISSN 0734189X. doi : 10.1016/0734-189X(85)90125-2.

[Karp, 1972] R. M. Karp. Reducibility among combinatorial problems. In *Complexity of Computer Computations*, pp. 85–103, 1972.

[Kaufmann, 1975] A. Kaufmann. *Introduction to the Theory of Fuzzy Subsets*. Academic Pr, New York, 1975. ISBN 0124023010.

[Kay, 1998] S.M. Kay. *Fundamentals of Statistical Signal Processing*, vol. 2 : Detection Theory. Prentice Hall Signal Processing Series, New Jersey, 1998.

[Kennedy & Eberhart, 1995] J. Kennedy & R. Eberhart. Particle swarm optimization. *IEEE International Conference on Neural Networks*, 4: 1942–1948, 1995.

[Kennedy & Mendes, 2002] J. Kennedy & R. Mendes. Population structure and particle swarm performance. *Computational Intelligence, Proceedings of the World on Congress on*, 2: 1671–1676, 2002.

[Kirkpatrick et al., 1983] S. Kirkpatrick, C. Gelatt, & M. Vecchi. Optimization by simulated annealing. *Science*, 220(4598): 671–680, 1983.

[Klir et al., 1997] G. J. Klir, U. St. Clair, & B. Yuan. *Fuzzy set theory : foundations and applications*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1997. ISBN 0-13-341058-7.

[Knuth, 1998] D. E. Knuth. *The art of computer programming, volume 3 : (2nd ed.) sorting and searching*. Addison Wesley Longman Publishing Co., Inc., Redwood City, CA, USA, 1998. ISBN 0-201-89685-0.

[Koza, 1992] J. R. Koza. *Genetic Programming : On the Programming of Computers by Means of Natural Selection (Complex Adaptive Systems)*. The MIT Press, 1 edition, December 1992. ISBN 0262111705.

[Koza, 1994] J. R. Koza. Introduction to genetic programming. In Kenneth E. Kinnear, Jr., editor, *Advances in Genetic Programming*, chapter 2, pp. 21–42. MIT Press, Cambridge, MA, USA, 1994.

[Koziel & Michalewicz, 1999] S. Koziel & Z. Michalewicz. Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization. *Evol. Comput.*, 7(1): 19–44, 1999. ISSN 1063-6560. doi : http://dx.doi.org/10.1162/evco.1999.7.1.19.

[Kuhn, 1982] H. W. Kuhn. Nonlinear programming : a historical view. *SIGMAP Bull.*, (31): 6–18, June 1982. ISSN 0163-5786. doi : 10.1145/1111278.1111279.

[Kuri-Morales & Gutiérrez-García, 2001] A. A. Kuri-Morales & J. Gutiérrez-García. Penalty Functions Methods for Constrained Optimization with Genetic Algorithms : A Statistical Analysis. In Carlos A. Coello Coello, Alvaro de Albornoz, Luis Enrique Sucar, & Osvaldo Cairó Battistutti, editors, *Proceedings of the 2nd Mexican International Conference on Artificial Intelligence (MICAI 2002)*, pp. 108–117, Heidelberg, Germany, April 2001. Mérida, Yucatán, México, Springer-Verlag. Lecture Notes in Artificial Intelligence Vol. 2313.

[Land & Doig, 1960] A. H. Land & A. G. Doig. An automatic method of solving discrete programming problems. *Econometrica*, 28(3): 497–520, 1960.

[Larrañaga & Lozano, 2002] P. Larrañaga & J. A. Lozano, editors. *Estimation of Distribution Algorithms : A New Tool for Evolutionary Computation*. Kluwer Academic Publishers, Boston, MA, 2002.

[Lawler et al., 1985] E. L. Lawler, Jan Karel Lenstra, Alexander H. G. Rinnooy Kan, & David B. Shmoys, editors. *The Traveling Salesman Problem*. John Wiley & Sons Ltd., Chichester, 1985.

[Levine & Nazif, 1985] M.D. Levine & A.M. Nazif. Dynamic measurement of computer generated image segmentations. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 7: 155–164, 1985.

[Levy, 2008] B. C. Levy. *Principles of Signal Detection and Parameter Estimation*. Springer Publishing Company, Incorporated, 2008. ISBN 0387765425, 9780387765426.

[Liang et al., 2005] J.J. Liang, P.N. Suganthan, & K. Deb. Novel composition test functions for numerical global optimization. In *Proceedings of Swarm Intelligence Symposium*, pp. 68–75, 2005.

[Ma, 2010] H. Ma. An analysis of the equilibrium of migration models for biogeography-based optimization. *Information Sciences*, 180(18): 3444–3464, 2010.

[Ma & Simon, 2011] Haiping Ma & Dan Simon. Blended biogeography-based optimization for constrained optimization. *Eng. Appl. of AI*, 24(3): 517–525, 2011.

[MacArthur & Wilson, 1967] R. MacArthur & E. Wilson. *The Theory of Biogeography*. Princeton University Press, Princeton, NJ, 1967.

[Marichal, 2002] J.-L. Marichal. Aggregation of interacting criteria by means of the discrete Choquet integral. In T. Calvo, G. Mayor, & R. Mesiar, editors, *Aggregation Operators : New Trends and Applications*, vol. 97 of *Studies in Fuzziness and Soft Computing*, pp. 224–244. Physica Verlag, Heidelberg, 2002.

[Metropolis et al., 1953] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, & E. Teller. Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, 21(6): 1087–1090, 1953.

[Mezura-Montes & Coello, 2005] E. Mezura-Montes & C. A. C. Coello. A simple multimembered evolution strategy to solve constrained optimization problems. *IEEE Trans. Evolutionary Computation*, 9(1): 1–17, 2005.

[Michalewicz, 1994] Z. Michalewicz. *Genetic algorithms + data structures = evolution programs (2nd, extended ed.)*. Springer-Verlag New York, Inc., New York, NY, USA, 1994. ISBN 3-540-58090-5.

[Michalewicz & Janikow, 1991] Z. Michalewicz & C. Z. Janikow. Handling constraints in genetic algorithms. In *Proceedings of the 4th International Conference on Genetic Algorithms*, pp. 151–157, San Diego, CA, USA, 1991.

[Michalewicz & Nazhiyath, 1995] Z. Michalewicz & G. Nazhiyath. Genocop iii : A co-evolutionary algorithm for numerical optimization problems with nonlinear constraints. In *D. B. Fogel (Ed.), Proceedings of the second IEEE International Conference on Evolutionary Computation*, pp. 647–651, Piscataway, NJ, 1995. IEEE Press.

[Michalewicz & Schoenauer, 1996] Z. Michalewicz & M. Schoenauer. Evolutionary Algorithms for Constrained Parameter Optimization Problems. *Evolutionary Computation*, 4(1): 1–32, 1996.

[Michalewicz et al., 1996] Z. Michalewicz, G. Nazhiyath, & M. Michalewicz. A note on usefulness of geometrical crossover for numerical optimization problems. In *Evolutionary Programming*, pp. 305–312, 1996.

[Mladenovic, 1995] N. Mladenovic. A variable neighborhood algorithm - a new metaheuristic for combinatorial optimization. In *Abstracts of papers presented at Optimization Days*, pp. 112, Montréal, Canada, May 1995.

[Mladenovic & Hansen, 1997] N. Mladenovic & P. Hansen. Variable neighborhood search. *Computers and Operations Research*, 24: 1097–1100, 1997.

[Mühlenbein & Paaß, 1996] H. Mühlenbein & G. Paaß. From recombination of genes to the estimation of distributions i. binary parameters. In *Proceedings of the 4th International Conference on Parallel Problem Solving from Nature*, PPSN IV, pp. 178–187, London, UK, 1996. Springer-Verlag. ISBN 3-540-61723-X.

[Mühlenbein et al., 1991] H. Mühlenbein, M. Schomisch, & J. Born. The parallel genetic algorithm as function optimizer. *Parallel Computing*, 17(6-7): 619–632, 1991.

[Neyman & Pearson, 1933] J. Neyman & E. S. Pearson. On the problem of the most efficient tests of statistical hypotheses. *Royal Society of London Philosophical Transactions Series A*, 231: 289–337, 1933.

[Noman & Iba, 2008] N. Noman & I. Iba. Accelerating differential evolution using an adaptive local search. *IEEE Trans. Evolutionary Computation*, 12(1): 107–125, 2008.

[Pal & Pal, 1992] N. R. Pal & S. K. Pal. Higher order fuzzy entropy and hybrid entropy of a set. *Inf. Sci.*, 61: 211–231, June 1992. ISSN 0020-0255.

[Pal & Pal, 1993] N. R. Pal & S. K. Pal. A review on image segmentation techniques. *Pattern Recognition*, 26(9): 1277–1294, September 1993. ISSN 00313203.

[Papadimitriou, 1994] C. H. Papadimitriou. *Computational complexity.* Addison-Wesley, 1994. ISBN 978-0-201-53082-7.

[Papadimitriou & Steiglitz, 1982] C. H. Papadimitriou & K. Steiglitz. *Combinatorial optimization : algorithms and complexity.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1982. ISBN 0-13-152462-3.

[Paredis, 1994] J. Paredis. Co-evolutionary constraint satisfaction. In *Proceedings of the International Conference on Evolutionary Computation. The Third Conference on Parallel Problem Solving from Nature : Parallel Problem Solving from Nature*, PPSN III, pp. 46–55, London, UK, 1994. Springer-Verlag. ISBN 3-540-58484-6.

[Pareto, 1896] V. Pareto. *Cours d'Economie Politique.* Droz, Genève, 1896.

[Parmee & Purchase, 1994] I. C. Parmee & G. Purchase. The development of a directed genetic search technique for heavily constrained design spaces. In I. C. Parmee, editor, *Adaptive Computing in Engineering Design and Control-'94*, pp. 97–102, Plymouth, UK, 1994. University of Plymouth.

[Passino, 2002] K. M. Passino. Biomimicry of Bacterial Foraging for distributed optimization and control. *IEEE Control Systems Magazine*, 22(3): 52–67, 2002. doi : 10.1109/MCS.2002.1004010.

[Poor, 1998] H.V. Poor. *An Introduction to Signal Detection and Estimation. Springer Texts in Electrical Engineering.* Springer 2nd edition, New York, 1998.

[Potter & De Jong, 1994] M. A. Potter & K. De Jong. A cooperative coevolutionary approach to function optimization. In *Proceedings of the International Conference on Evolutionary Computation. The Third Conference on Parallel Problem Solving from Nature : Parallel Problem Solving from Nature*, PPSN III, pp. 249–257, London, UK, 1994. Springer-Verlag. ISBN 3-540-58484-6.

[Price et al., 2005] Kenneth V. Price, Rainer M. Storn, & Jouni A. Lampinen. *Differential Evolution A Practical Approach to Global Optimization.* Natural Computing Series. Springer-Verlag, Berlin, Germany, 2005.

[Pulido & Coello, 2004] G. T. Pulido & C. A. C. Coello. A constraint-handling mechanism for particle swarm optimization. In *Congress on Evolutionary Computation*, vol. 2, pp. 1396–1403, 2004.

[Pun, 1980] T Pun. A new method for gray-level picture thresholding using the entropy of histogram. *Signal Processing*, 2(3): 223–237, 1980.

[Pun, 1981] T Pun. Entropic thresholding, a new approach. *Computer Graphics and Image Processing*, 16(3): 210–239, 1981.

[Rechenberg, 1965] I. Rechenberg. Cybernetic solution path of an experimental problem. Technical report, Royal Air Force Establishment, 1965.

[Rechenberg, 1973] I. Rechenberg. *Evolutionsstrategie : optimierung technischer systeme nach prinzipien der biologischen evolution.* Frommann-Holzboog, Stuttgart, 1973.

[Reynolds, 1987] C. W. Reynolds. Flocks, herds, and schools : A distributed behavioral model. In *Computer Graphics*, vol. 21, pp. 25–34, 1987.

[Reynolds, 1994] R. G. Reynolds. An Introduction to Cultural Algorithms. In A. V. Sebalk & L. J. Fogel, editors, *Proceedings of the Third Annual conference on Evolutionary Programming*, pp. 131–139, River Edge NJ, 1994. World Scientific Publishing.

[Ricklefs & Miller, 2005] R. Ricklefs & G. Miller. *Ecologie.* éd. de Boeck, 2005. ISBN 2-7445-0145-X.

[Ritzel et al., 1994] B. J. Ritzel, J. W. Eheart, & S. Ranjithan. Using genetic algorithms to solve a multiple objective groundwater pollution containment problem. *Water Resources Researc*, 30(5): 1589–1603, 1994.

[Rosenbrock, 1960] H. H. Rosenbrock. An Automatic Method for Finding the Greatest or Least Value of a Function. *The Computer Journal*, 3(3): 175–184, January 1960. doi : 10.1093/comjnl/3.3.175.

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

[Sahoo et al., 1988] P. K. Sahoo, S. Soltani, A. K.C. Wong, & Y. C. Chen. A survey of thresholding techniques. *Comput. Vision Graph. Image Process.*, 41: 233–260, February 1988. ISSN 0734-189X.

[Sahoo et al., 1997] P. K. Sahoo, C. Wilkins, & J. Yeager. Threshold selection using reny's entropy. *Pattern Recognition*, 30(i1): 71–84, 1997.

[Schaffer, 1985] J. D. Schaffer. Multiple objective optimization with vector evaluated genetic algorithms. In *Proceedings of the 1st International Conference on Genetic Algorithms*, pp. 93–100, Hillsdale, NJ, USA, 1985. L. Erlbaum Associates Inc. ISBN 0-8058-0426-9.

[Schoenauer & Michalewicz, 1996] M. Schoenauer & Z. Michalewicz. Evolutionary computation at the edge of feasibility. In *Parallel Problem Solving from Nature - PPSN IV, International Conference on Evolutionary Computation. The 4th International Conference on Parallel Problem Solving from Nature*, pp. 245–254, Berlin, Germany, 1996.

[Schoenauer & Michalewicz, 1997] M. Schoenauer & Z. Michalewicz. Boundary operators for constrained parameter optimization problems. In *Proceedings of the 7th International Conference on Genetic Algorithms*, pp. 322–329, East Lansing, MI, USA, 1997.

[Schoenauer & Xanthakis, 1993] M. Schoenauer & S. Xanthakis. Constrained ga optimization. In *Proceedings of the 5th International Conference on Genetic Algorithms*, pp. 573–580. Morgan Kaufmann, 1993.

[Schwefel, 1981] Hans-Paul Schwefel. *Numerical Optimization of Computer Models*. John Wiley & Sons, Inc., New York, NY, USA, 1981. ISBN 0471099880.

[Sezgin & Sankur, 2004] Mehmet Sezgin & Bülent Sankur. Survey over image thresholding techniques and quantitative performance evaluation. *Journal of Electronic Imaging*, 13 (1): 146–168, 2004.

[Shafer, 1976] G. Shafer. *A mathematical theory of evidence*. Princeton university press, 1976.

[Shannon, 1948] C. E. Shannon. *A Mathematical Theory of Communication*. CSLI Publications, 1948.

[Shi & Eberhart, 1998] Y. Shi & R.C. Eberhart. A modified particle swarm optimizer. In *Proceedings of IEEE International Conference on Evolutionary Computation*, pp. 69–73, Washington, DC, USA, May 1998. IEEE Computer Society.

[Simon, 2008] D. Simon. Biogeography-based optimization. *IEEE Transactions on Evolutionary Computation*, 12(6): 702–713, 2008.

[Smets, 1996] P. Smets. Imperfect information : Imprecision and uncertainty. In *Uncertainty Management in Information Systems*, pp. 225–254. 1996.

[Smith & Tate, 1993] A. E. Smith & D. M. Tate. Genetic optimization using a penalty function. In *Proceedings of the 5th International Conference on Genetic Algorithms*, pp. 499–505, Urbana-Champaign, IL, USA, 1993.

[Srinivas & Deb, 1994] N. Srinivas & K. Deb. Muiltiobjective optimization using nondominated sorting in genetic algorithms. *Evol. Comput.*, 2(3): 221–248, September 1994. ISSN 1063-6560. doi : 10.1162/evco.1994.2.3.221.

[Stanley & Miikkulainen, 2004] K. O. Stanley & R. Miikkulainen. Competitive coevolution through evolutionary complexification. *Journal of Artificial Intelligence Research*, 21 (1): 63–100, 2004.

[Storn & Price, 1997] R. M. Storn & K. V. Price. Differential evolution a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4): 341–359, December 1997. ISSN 0925-5001. doi : 10.1023/A:1008202821328.

[Stützle, 1998] T. Stützle. *Local Search Algorithms for Combinatorial Problems : Analysis, Improvements, and New Applications*. Phd thesis, Darmstadt University of Technology, 1998.

[Surry et al., 1995] P. D. Surry, N. J. Radcliffe, & I. D. Boyd. A Multi-Objective Approach to Constrained Optimisation of Gas Supply Networks : The COMOGA Method. In Terence C. Fogarty, editor, *Evolutionary Computing. AISB Workshop. Selected Papers*, pp. 166–180, Sheffield, U.K., April 1995. Springer-Verlag. Lecture Notes in Computer Science No. 993.

[Taillard, 2002] É. D. Taillard. Principes d'implémentation des métaheuristiques". In Jacques Teghem & Marc Pirlot, editors, *Optimisation approchée en recherche opérationnelle*, pp. 57–79. Hermès, 2002.

[Talbi, 2009] E.-G. Talbi. *Metaheuristics : From Design to Implementation*. Wiley-Blackwell, first edition, 2009.

[Tao et al., 2003] W. Tao, J.-W. Tian, & J. Liu. Image segmentation by three-level thresholding based on maximum fuzzy entropy and genetic algorithm. *Pattern Recogn. Lett.*, 24 (16): 3069–3078, December 2003. ISSN 0167-8655. doi : 10.1016/S0167-8655(03)00166-1.

[Tao et al., 2007] W. Tao, H. Jin, & L. Liu. Object segmentation using ant colony optimization algorithm and fuzzy entropy. *Pattern Recogn. Lett.*, 28(7): 788–796, May 2007. ISSN 0167-8655. doi : 10.1016/j.patrec.2006.11.007.

[Tenny & Sandell, 1981] R. R. Tenny & N. R. Sandell. Detection with distributed sensors. *IEEE Transactions on Aerospace and Electronic Systems*, 17: 501–510, 1981.

[Timmis et al., 2008] J. Timmis, P. Andrews, N. Owens, & E. Clark. An interdisciplinary perspective on artificial immune systems. *Evolutionary Intelligence*, 1(1): 5–26, 2008.

[Tobias & Seara, 2002] O.J. Tobias & R. Seara. Image segmentation by histogram thresholding using fuzzy sets. 11(12): 1457–1465, December 2002.

[Törn & Zilinskas, 1989] A. A. Törn & A. Zilinskas. *Global Optimization*, vol. 350 of *Lecture Notes in Computer Science*. Springer, 1989. ISBN 3-540-50871-6.

[Trees, 1998] H.L.V. Trees. *Detection, Estimation, and Modulation Theory*. Wiley-Interscience, New York, 1998.

[Tsallis, 1988] C. Tsallis. Possible generalization of Boltzmann-Gibbs statistics. *J. Stat. Physics*, 52, 1988.

[Tsitsiklis, 1993] J.N. Tsitsiklis. Decentralized detection. *Advances in Statistical Signal Processing*, 2: 297–344, 1993.

[Tylor, 1924] E. B. Tylor. *Primitive Culture*, vol. 2 vols. 7th edition, New York : Brentano's, 1924.

[Waagen et al., 1992] D. Waagen, P. Diercks, & J. McDonnell. The stochastic direction set algorithm : a hybrid technique for finding function extrema. In D. B. Fogel & W. Atmar (Eds.), editor, *Proceedings of the First Annual Conference on Evolutionary Programming*, pp. 35–42, La Jolla, CA, 1992. Evolutionary Programming Society.

[Walker et al., 1993] A. Walker, J. Hallam, & D. Willshaw. Bee-havior in a Mobile Robot : The Construction of a Self-Organized Cognitive Map and its Use in Robot Navigation within a Complex, Natural Environment. In *Proc. ICNN'93, Int. Conf. on Neural Networks*, vol. III, pp. 1451–1456, Piscataway, NJ, 1993. IEEE Service Center.

[Wimalajeewa & Jayaweera, 2008] T. Wimalajeewa & S. K. Jayaweera. Optimal power scheduling for correlated data fusion in wireless sensor networks via constrained pso. *Trans. Wireless. Comm.*, 7(9): 3608–3618, September 2008. ISSN 1536-1276. doi : 10.1109/TWC.2008.070386.

[Wolpert & Macready, 1997] D. H. Wolpert & W. G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1): 67–82, April 1997. ISSN 1089-778X.

[Yao et al., 1999] X. Yao, Y. Liu, & G. Lin. Evolutionary programming made faster. *IEEE Transactions on Evolutionary Computation*, 3: 82–102, 1999.

[Zadeh, 1965] L.A. Zadeh. Fuzzy sets. *Information and Control*, 8: 338–353, 1965.

[Zhang, 1996] Y. J. Zhang. A survey on evaluation methods for image segmentation. *Pattern Recognition*, 29(8): 1335–1346, August 1996. ISSN 00313203. doi : 10.1016/0031-3203(95) 00169-7.

[Zhao et al., 2001] M. Zhao, A.M.N. Fu, & H. Yan. A technique of three-level thresholding based on probability partition and fuzzy 3-partition. *IEEE Transactions on Fuzzy Systems*, 9(3): 469–479, 2001.

[Zimmermann, 1991] H.J. Zimmermann. *Fuzzy Set Theory and its applications*. Kluwer Academic, Dordrecht, 1991.

[Zucker, 1976] S. W. Zucker. Region growing : Childhood and Adolescence. *Computer Graphics and Image Processing*, 5(3): 382–399, September 1976.

Index