Amélioration des métaheuristiques d'optimisation à l'aide de l'analyse de sensibilité

Peio Loubiere

UNIVERSITÉ PARIS-EST

École Doctorale 532 Mathématiques et Sciences et Technologies de l'Information et de la Communication (MSTIC)
Laboratoire Images, Signaux et Systèmes Intelligents (EA 3956)

Thèse

présentée pour l'obtention du grade de DOCTEUR
DE L'UNIVERSITE PARIS-EST par

Peio Loubière

Amélioration des métaheuristiques d'optimisation à l'aide de l'analyse de sensibilité

Soutenue le 21 novembre 2016, devant le jury composé de :

Laurent DUMAS	Professeur des Universités	Université de Versailles	Président du jury
Stéphane CANU	Professeur des Universités	INSA Rouen	Rapporteur
André ROSSI	Professeur des Universités	Université d'Angers	Rapporteur
Nicolas MONMARCHÉ	Maître de Conférences	Université de Tours	Examinateur
Astrid JOURDAN	Enseignante-chercheuse	EISTI Pau	Co-Encadrante
Rachid CHELOUAH	Directeur de recherche	EISTI Cergy-Pontoise	Co-Directeur de thèse
Patrick SIARRY	Professeur des Universités	Université de Paris-Est Créteil	Directeur de thèse

---

Remerciements

Je remercie énormément Patrick Siarry de l'Université de Paris-Est Créteil et Rachid Chelouah de l'EISTI, Co-Directeurs de cette thèse, qui ont ouvert la porte à mes velléités estudiantines. L'encadrement d'un jeune vieux thésard doit être un exercice assez périlleux ; leur soutien indéfectible, leur allant, leur disponibilité, leurs encouragements, leurs remarques, leur entrain, leur bienveillance et leur patience ont été autant de bouées dans l'océan d'émotions que j'ai traversé durant ces trois années. Merci, incroyablement.

Je souhaite également exprimer mon immense gratitude à Astrid Jourdan qui a co-encadré et coloré cette thèse de mathématiques et de statistiques. Malgré les nombreuses tentatives de navigation entre les Charybde et Scylla des Statistiques au cours de ma scolarité, j'ai finalement accosté le continent et m'y suis promené avec sérénité et presque délectation. La boucle est bouclée, merci !

Je tiens aussi à remercier Nesim Fintz, directeur général fondateur de l'EISTI, de m'avoir soutenu et financé durant ces trois années et dont j'espère avoir honoré l'aide apportée et la confiance investie.

Merci encore aux rapporteurs de cette thèse, Stéphane Canu et à André Rossi dont l'intérêt et la bienveillance portés m'ont permis d'améliorer ce travail. Merci à Laurent Dumas d'avoir accepté de présider mon jury ainsi qu'à Nicolas Monmarché, d'avoir accepté d'examiner mes travaux. À tous, un grand merci pour m'avoir fait l'honneur de composer mon jury.

Je remercie également chaleureusement Yacine Amirat, Sylvie Cach de l'ED MSTIC, ainsi que Mmes Patricia Jamin, Katia Lambert et Isabelle Villaumé qui par leur gentillesse, leur disponibilité et leur compréhension ont facilité ma situation particulière d'étudiant en exil.

Merci aussi au laboratoire Latep de l'ENS-GTI de Pau, à Erwin Franquet et Stéphane Gibout qui m'ont permis d'éprouver mes méthodes sur un cas concret et fait preuve de beaucoup de sympathie et de patience. Je remercie également Pierre Cézac et Jean-Michel Reneaume pour leur amicale

participation et leur bienveillance et dont j'espère pouvoir combler les ambitions !

Je remercie tous mes collègues de l'EISTI qui m'ont soutenu, détendu, relu, fait-des-emplois-du-temps-super-sympas, supporté, dépanné, consolé et conseillé. Je remercie particulièrement Florent Devin et Yannick Le Nir, qui m'ont fait confiance dès le début de ma nouvelle vie et qui ont allumé la mèche. Malgré les manqués, on a fini par y arriver !

Merci à feux-(L@ris, LISMMA, Ecs-Lab) et au Laboratoire Quartz, notamment à Zoubeida Abdelmoula, Christel Compagnon, Jean-Yves Choley et Jean-Pierre Barbot qui ont maintenu une motivante émulation. Merci également à Dominique Laffly et aux membres du projet TORUS, qui ouvrent la fenêtre des possibles. C'est enfin le début de l'histoire pour moi !

Merci aux amis de Lagunt eta Maita, l'autre remue-méninges, bulle d'air nécessaire. L'année prochaine, je m'y mets sérieusement, *Hitza hitz !* De gigantesques mercis, pleins de fourmis dans les jambes et de « tardance » à tous mes amis, pour leur soutien inconditionnel et leur compréhension rassurante : – « *Attendez-moi, j'arrive !* » ; à ma famille, aimante, et en particulier à la mémoire de Georgette Curutchet et de Maïté Sicre ; enfin, à Mélanie dont l'amour, l'humour et l'énergie ont pavé la route de brique jaune conduisant à ce travail, *"There's no place like home"*.

Je dédie mon travail à mes parents, *maite zaituztet, betiko.*

La bande originale de cette thèse a été composée par les Swans, Joy Division, Deutsch Nepal, Sieben, Hint, Death In June, Godspeed You ! Black Emperor, Nancy Sinatra, Crimson Scarlet, Taake, Dälek, Sonic Youth, The Cramps, Nick Cave & The Bad Seeds, The Stooges, Fela, Yann Tiersen, Frustration, Utarm, Scott Walker, White Zombie, Public Enemy, Lustmord, Chet Baker, Bauhaus, Wardruna, The Limiñanas, Savage Republic, Beastie Boys, Sham 69, Dead Can Dance, Seven That Spells, Faith No More, Eyehategod, The Ramones, Dead boys, Suicidal Tendencies, Sick Of It All, Nico, Splintered, Portishead, Godflesh, Keith Jarrett, Trepanenringsritualen, Barcode, Nina Simone, Sleaford Mods, Primus, The Misfits, Sol Invictus, Anthrax, Bad Religion, Unsane, Mueran Humanos, The Black Angels, Jastreb, Desiderii Marginis, Control, Neurosis, Richard Cheese, Inter Arma, The Young Gods, f/i, Stella, Massive Attack, Test Dept, Electric Wizard, Coil, Les Wampas, Stereolab, Les Thugs, Nadja...

"Searching, seek and destroy !"

James Hetfield, *Seek & destroy*

Résumé

L'*optimisation difficile* représente une classe de problèmes dont la résolution ne peut être obtenue par une méthode exacte en un temps polynomial. Trouver une solution en un temps raisonnable oblige à trouver un compromis quant à son exactitude. Les *métaheuristiques* sont une classe d'algorithmes permettant de résoudre de tels problèmes, de manière générique et *efficiente* (i.e. trouver une solution satisfaisante selon des critères définis : temps, erreur, etc.). Le premier chapitre de cette thèse est notamment consacré à la description de cette problématique et à l'étude détaillée de deux familles de métaheuristiques à population, les algorithmes évolutionnaires et les algorithmes d'intelligence en essaim.

Ce premier chapitre présente également la notion d'*analyse de sensibilité*. L'analyse de sensibilité permet d'évaluer *l'influence des variables* d'une fonction sur son résultat. Son étude caractérise globalement le comportement de la fonction à optimiser (linéarité, influence, corrélation, etc.) sur son espace de recherche.

L'incorporation d'une méthode d'analyse de sensibilité au sein d'une métaheuristique permet d'orienter sa recherche dans les directions les plus prometteuses. Deux algorithmes réunissant ces notions sont proposés aux deuxième et troisième chapitres. Pour le premier algorithme, ABC-Morris, la méthode d'analyse de sensibilité de Morris est introduite dans la métaheuristique de colonie d'abeilles artificielles (ABC).

Ces deux méthodes reposent sur des équations similaires, basées sur un déplacement unidimensionnel. Cela fait d'ABC-Morris un cas particulier. Afin de généraliser l'approche, une nouvelle méthode d'analyse de sensibilité, NN-LCC, est ensuite développée et la manière de l'intégrer au sein d'une métaheuristique est illustrée sur deux exemples, ABC avec taux de modification et évolution différentielle. L'efficacité des approches proposées est testée sur le jeu de données de la conférence CEC 2013.

Enfin, nous présentons une application de ces algorithmes à la caractérisation de matériaux à changement de phase. Nous exposons le contexte, ainsi que les questions posées pour l'adaptation correcte de nos méthodes, et les solutions proposées.

Mots clés : *Optimisation, optimisation continue, métaheuristiques, algorithmes évolutionnaires, évolution différentielle, intelligence en essaim, optimisation par essaim particulaire, colonies d'abeilles artificielles, analyse de sensibilité, corrélation linéaire, méthode de Morris, indices de sensibilité.*

Abstract

Hard optimization stands for a class of problems which solutions cannot be found by an exact method, with a *polynomial complexity*. Finding the solution in an acceptable time requires compromises about its accuracy. *Metaheuristics* are high-level algorithms that solve these kinds of problems. They are generic and *efficient* (i.e. they find an acceptable solution according to defined criteria such as time, error, etc.). The first chapter of this thesis is partially dedicated to the state-of-the-art of these issues, especially the study of two families of population based metaheuristics: evolutionary algorithms and swarm intelligence based algorithms.

Sensitivity analysis is also presented in this chapter. *Sensitivity analysis* aims at evaluating *variables' influence* on a function response. It globally characterizes an objective function behavior (linearity, non linearity, influence, etc.), over its search space.

Including a sensitivity analysis method in a metaheuristic enhances its search capabilities along most promising directions. Two algorithms, binding these two concepts, are proposed in the second and third chapters. In the first one, ABC-Morris, Morris sensitivity analysis method is included in artificial bee colony algorithm. This encapsulation is dedicated because of the similarity of their bare bone equations, based on an unidimensional offset. With the aim of generalizing the approach, a new sensitivity analysis method, NN-LCC, is developped and its generic integration is illustrated on two metaheuristics. The efficiency of both methods is tested on the CEC 2013 conference benchmark.

Finally we present an application of these algorithms to the identification of the enthalpy of phase change materials. We describe all the issues for adapting correctly our algorithms to the problem and the proposed answers.

Keywords : *Optimization, continuous optimization, metaheuristics, evolutionary algorithms, differential evolution, swarm intelligence, particle swarm optimization, artificial bee colony, sensitivity analysis, linear correlation, Morris method, sensitivity indices.*

Table des matières

Introduction générale **1**

1 État de l'art en optimisation difficile et analyse de sensibilité **5**

1.1 Généralités sur l'optimisation et les métaheuristiques 6

1.2 Algorithmes évolutionnaires 8

1.2.1 Algorithme Génétique 8

1.2.2 Évolution Différentielle 10

1.3 L'intelligence en essaim 19

1.3.1 Algorithme de Colonies de Fourmis 20

1.3.2 Algorithme d'Optimisation par Essaim Particulaire 24

1.3.3 Algorithme de Colonie d'Abeilles Artificielles 31

1.4 Généralités sur l'analyse de sensibilité 39

1.5 Méthodes basées sur la régression linéaire 42

1.6 Méthodes pas-à-pas (*One-At-Time*) 43

1.7 Méthode des Effets Élémentaires (la méthode de Morris) 44

1.8 Méthodes basées sur l'analyse de la variance 46

1.8.1 Les indices de Sobol 46

1.8.2 La méthode FAST 48

1.8.3 Méthode basée sur la construction de métamodèle 48

1.9 Conclusion 49

2 Élaboration de l'algorithme ABC-Morris pour un parcours guidé de l'espace de recherche **53**

2.1 Introduction 54

2.2 Description de notre variante de l'algorithme ABC 55

2.3 Tests et discussion 58

2.3.1 Fonctions de test utilisées pour la validation 59

2.3.2 Algorithmes utilisés . . . 59
2.3.3 Protocole de test . . . 60
2.3.4 Résultats et discussion . . . 60
2.3.5 Comparaison pour 100% de dimensions actives . . . 61
2.3.6 La question de la rotation . . . 69
2.4 Conclusion . . . 72

3 Élaboration de la méthode d'analyse de sensibilité NN-LCC pour les métaheuristiques à déplacement multidimensionnel 75
3.1 Introduction . . . 76
3.2 Méthodes d'analyse de sensibilité des plus proches voisins . . . 77
3.2.1 Description de l'algorithme . . . 77
3.2.2 Illustration . . . 79
3.2.3 Tests et discussion . . . 80
3.2.4 Test de la méthode d'analyse de sensibilité . . . 80
3.3 Intégration de la méthode aux métaheuristiques . . . 85
3.3.1 Intégration de la méthode dans l'algorithme ABCMR . . . 86
3.3.2 Intégration de la méthode dans l'algorithme DE . . . 86
3.4 Tests et résultats . . . 87
3.4.1 Paramétrage . . . 87
3.4.2 Résultats . . . 87
3.5 Conclusion . . . 91

4 Application à l'utilisation de méthodes inverses pour la caractérisation de matériaux à changement de phase 93
4.1 Présentation . . . 94
4.1.1 Contexte . . . 94
4.1.2 Calorimétrie différentielle à balayage . . . 96
4.2 Le problème d'optimisation : méthodes d'inversion pour la calorimétrie . . . 98
4.3 Mise en place de la solution . . . 98
4.3.1 Variables et paramètres . . . 99
4.3.2 Fonction objectif et solution . . . 99
4.3.3 Métaheuristique . . . 100
4.4 Discussion . . . 102
4.4.1 Dépendance entre variables . . . 102

4.4.2 Divergence du modèle 102
4.5 Conclusion 104

Conclusion générale et perspectives **105**

Annexes **109**

A Résultats du test de Wilcoxon 109
B Figures d'évolution de l'erreur médiane pour qABC et qABCMorris 114
C Scores des variables d'entrée pour NN-LCC et Morris 119
D Figures d'évolution de l'erreur médiane pour DE et DE-NN-LCC 122

Références bibliographiques **135**

Introduction générale

Le domaine de l'optimisation propose un ensemble de problématiques diverses et motivantes pour la communauté scientifique qui se doit de les résoudre de manière efficace, grâce à des méthodes innovantes de résolution. Ce domaine est transverse et agrège des disciplines variées telles que la logistique, l'électronique, le transport, la planification, le réseau et bien plus encore.

Résoudre un problème d'optimisation, pour un ensemble de données (ou variables), revient à trouver la meilleure solution selon un critère d'évaluation, donné par une fonction de coût (dite fonction objectif), que l'on souhaite minimiser ou maximiser. Chacune des variables est définie sur un domaine (ou espace) de recherche, généralement borné. Un problème peut être composé de variables appartenant à un domaine de recherche discret (dénombrable), continu (non dénombrable) ou encore aux deux, on parle alors de problème à variables mixtes (composé de variables discrètes et continues). La fonction objectif peut définir un unique critère d'optimisation (par exemple, le problème du voyageur de commerce [1], on dit que le problème est mono-objectif. Elle peut également en spécifier plusieurs, souvent contradictoires (trouver le chemin le plus rapide et le moins cher), on parle alors d'optimisation multi-objectifs. La fonction de coût, le domaine de recherche peuvent par ailleurs être modifiés au cours du temps (ajout ou suppression d'une ville à rejoindre sur le trajet). Dans ce cas, le problème est dit d'optimisation dynamique. Enfin, dans le cas où une solution quelconque du problème possède des contraintes de faisabilité (par exemple, il n'existe pas de route reliant directement deux villes), on parle d'optimisation sous contraintes.

Parmi l'ensemble des méthodes d'optimisation existantes, les métaheuristiques résolvent les problèmes de manière générique. Ces algorithmes permettent d'agréger différentes instances de problèmes sans modification fondamentale de leur fonctionnement. Ils sont essentiellement utilisés pour des problèmes d'optimisation dite « difficile » (cette notion n'est pas précisément définie et dépend de l'état de l'art en matière de méthodes de résolution), pour lesquels il est actuellement impossible de garantir la meilleure solution en un temps raisonnable. Ils fournissent une solution approchée

1. Étant donnée une liste de villes et les distances séparant chacune d'entre elles, quelle est la plus courte tournée possible qui visite chaque ville une seule fois et revient au point de départ ?

acceptable en déplaçant un ensemble de solutions (ou population) selon diverses stratégies ; ils ont en commun le fait de posséder une composante aléatoire. Les métaheuristiques les plus populaires sont basées sur des analogies avec la biologie (algorithmes évolutionnaires), la physique (recuit simulé) ou l'éthologie (essaim particulaire, colonie d'abeilles artificielles).

La plupart des axes de recherche proposés dans la littérature se concentrent sur la compréhension du comportement des métaheuristiques. Ils étudient leur fonctionnement intrinsèque, le calibrage convenable de leurs paramètres, leur auto-adaptation. Une problématique régulièrement approfondie est l'équilibre entre les techniques de maintien de la diversité des solutions et du parcours du domaine de recherche, pour trouver des zones prometteuses (où se situent de bonnes solutions) et les techniques de recherche locale, autour d'une bonne solution, pour favoriser la convergence. Différentes stratégies sont employées, telles que :

— la modification des équations de déplacement de leur population ;
— le partitionnement de l'espace de recherche et la résolution en sous-problèmes ;
— le partage d'informations et la collaboration en structure de voisinage de solutions ;
— l'utilisation d'une mémoire des bonnes solutions trouvées au cours du parcours ;
— l'hybridation avec des techniques de recherche locales ou entre métaheuristiques.

Dans ce travail, une approche, non pas centrée sur la métaheuristique, mais sur la fonction de coût est proposée. Le principe développé à travers nos travaux est d'utiliser l'information sur la fonction objectif acquise au cours du déroulement de l'algorithme d'optimisation, afin de quantifier l'influence des variables d'un problème sur l'évaluation de la fonction objectif. L'adaptation de méthodes d'analyse de sensibilité permet une telle étude. De manière générale, l'analyse de sensibilité est utilisée en tant que pré-étude, lorsque l'on a à disposition un code de calcul dont on ne possède pas l'expression analytique, et que l'on souhaite connaître l'influence des variables en entrée et réaliser une réduction de dimensions du problème. Elle permet également, dans le contexte des métaheuristiques, d'évaluer l'effet du paramétrage d'un algorithme sur son résultat afin de le calibrer correctement.

Une analyse de sensibilité est généralement basée sur un échantillonnage de points sur le domaine de recherche et sur différentes techniques permettant de quantifier cette influence, ou au moins de détecter les variables (i.e. dimensions) du problème qui n'ont pas ou très peu d'influence. Nous nous attachons ici à l'incorporer au sein d'une métaheuristique en utilisant les différentes évaluations réalisées durant son déroulement. Cette étude effectuée, l'objectif est alors de l'utiliser au cours de l'algorithme lors du déplacement, de la génération d'une solution, en favorisant les dimensions les plus intéressantes à explorer, selon leur influence.

Nous présentons un algorithme illustrant cette méthodologie ainsi qu'une nouvelle méthode d'analyse de sensibilité exploitable par différentes typologies de métaheuristiques.

Cette thèse a été effectuée dans le cadre de l'école doctorale MSTIC (E.D. 532) de l'Université Paris-Est. Elle a été financée par un C.D.I. à l'École Internationale des Sciences et du Traitement de l'Information (EISTI) et rendue possible par une décharge d'enseignement. Cette thèse a été co-dirigée par le Professeur P. Siarry, Directeur du groupe Signal, Image et Optimisation du Laboratoire Images, Signaux et Systèmes Intelligents (LiSSi, E.A. 3956, Université de Paris-Est Créteil, UPEC) et Rachid Chelouah, Enseignant-chercheur, HDR à l'EISTI, Cergy-Pontoise. Elle a également été co-encadrée par Astrid Jourdan, Enseignante-chercheuse à l'EISTI, Pau.

Les principales contributions de ce travail sont l'introduction d'une étude du comportement de la fonction de coût à optimiser par une méthode d'analyse de sensibilité, afin de guider la métaheuristique utilisée dans son domaine de recherche, ainsi que la création d'une nouvelle méthode d'analyse de sensibilité permettant d'étendre le nombre de métaheuristiques pouvant profiter de cette étude.

Cette thèse s'articule autour du plan suivant :

Le chapitre 1 présente les problèmes d'optimisation, les métaheuristiques, puis s'attarde sur un état de l'art de plusieurs métaheuristiques à population, classées en deux familles : les algorithmes évolutionnaires et les algorithmes de la famille d'intelligence en essaim. Cette partie présente plus particulièrement trois algorithmes dédiés à l'optimisation à variables continues : l'évolution différentielle (DE), l'optimisation par essaim particulaire (PSO) et l'algorithme de colonie d'abeilles artificielles (ABC). Une bibliographie sur les différents travaux d'amélioration est présentée. Elle s'appuie sur les différents points caractéristiques de chaque algorithme, comme la génération d'un nouvel individu, le partage d'information en voisinage ou l'auto-adaptation. Ces trois algorithmes ont été retenus car ils présentent chacun un comportement particulier dans la création d'un nouvel individu. ABC effectue une modification unidimensionnelle d'une solution existante, DE réalise une modification pour un sous-ensemble de dimensions tandis que PSO réalise une modification de l'ensemble des dimensions. Ces types de comportement sont à la base des choix proposés par la suite.

Nous proposons également dans ce chapitre une présentation de l'analyse de sensibilité. Les principales méthodes sont décrites ; elles peuvent être locales ou globales, basées sur la régression linéaire, les effets élémentaires ou basées sur l'analyse de la variance. Cette partie pose les bases des concepts utilisés ainsi que les contraintes qui ont orienté nos choix.

Le premier algorithme réalisé, ABC-Morris, exposé au chapitre 2, est à la base de notre contribution. Il introduit le principe d'utilisation d'une méthode d'analyse de sensibilité au sein d'une

métaheuristique, afin d'acquérir de l'information sur le comportement d'une fonction. Une mesure d'influence sur le résultat de la fonction objectif est définie pour chaque dimension. L'analyse de sensibilité permet de calculer cette mesure, en utilisant les évaluations réalisées par la métaheuristique durant son exécution. Nous présentons alors un algorithme ABC modifié, intégrant la méthode de Morris. La justification du choix de l'hybridation des deux méthodes, de l'algorithme de calcul de l'influence ainsi que la modification de la prise de décision de la métaheuristique sont détaillées. La pertinence de l'algorithme présenté est testée sur le jeu de données de la conférence *Conference on Evolutionary Computation* (CEC) 2013.

Une généralisation du principe précédent est exposée au chapitre 3. L'objectif est de proposer le concept d'étude du comportement d'une fonction au-delà du cas particulier de l'algorithme ABC. La méthode doit pouvoir détecter l'influence de chaque dimension, en utilisant toujours les évaluations fournies par une métaheuristique. Elle doit s'étendre aux cas où la création d'un individu correspond à un déplacement multi-dimensionnel. Nous présentons alors, dans un premier temps une méthode d'analyse hybride s'inspirant de la régression linéaire et de la méthode de Morris. Le comportement de cette méthode est testé sur les fonctions usuelles de l'analyse de sensibilité, puis son intégration est évaluée pour deux métaheuristiques différentes : ABC avec taux de modification et DE. Les algorithmes ainsi définis sont testés sur le jeu de données de la conférence CEC 2013.

Le chapitre 4 présente une application de nos méthodes à la caractérisation de matériaux à changement de phase. Cette caractérisation est réalisée par méthode inverse. Cette dernière consiste à évaluer les paramètres d'un phénomène physique à partir d'une valeur observée lors d'une expérience. C'est un problème d'optimisation pour lequel on essaie de minimiser l'erreur quadratique entre cette valeur observée et les valeurs obtenues par simulation numérique. L'exécution d'un code de calcul étant coûteux, il est intéressant d'éprouver nos méthodes permettant de détecter les variables peu influentes, afin de trouver une solution acceptable rapidement.

Enfin, une conclusion générale clôt ce manuscrit. Elle reprend les contributions effectuées et présente les limites et les perspectives d'amélioration envisagées.

Chapitre 1

État de l'art en optimisation difficile et analyse de sensibilité

Il s'agit de présenter ici un ensemble de concepts relatifs à l'optimisation difficile. Plus précisément, cette partie contient un état de l'art autour des problèmes d'optimisation continue, mono-objectif, des termes utilisés et des différentes approches de résolution par métaheuristique.

Il s'agit également de présenter l'analyse de sensibilité, ses objectifs ainsi que la méthodologie associée. Nous présentons les principales méthodes existantes et nous nous demanderons dans quelle mesure le fonctionnement d'une métaheuristique peut tirer bénéfice d'une analyse de sensibilité au cours de son exécution.

Notions abordées dans le chapitre :

- *Problèmes d'optimisation statique, mono-objectif, continue ;*
- *Métaheuristiques à population ;*
- *Algorithmes évolutionnaires et de la famille d'intelligence en essaim ;*
- *Régression linéaire ;*
- *Méthode des effets élémentaires, méthode de Morris ;*
- *Indices de sensibilité.*

1.1 Généralités sur l'optimisation et les métaheuristiques

Un problème d'optimisation est défini comme la recherche, dans un espace de solutions, d'une solution optimale quantifiée par une fonction objectif. Cette quantification conduit à vouloir maximiser ou minimiser le problème. Un ensemble de méthodes exactes permet de trouver une solution en un temps fini et, de manière générale, polynomial. Ces méthodes, parmi lesquelles on peut citer la méthode du Gradient, la méthode de Newton, la méthode Séparation et Évaluation (*Branch and Bound*) etc., sont déterministes : elles fournissent le meilleur résultat attendu en un nombre fini d'étapes, par dérivation ou par parcours de tout ou partie des solutions.

L'optimisation difficile représente une classe de problèmes d'optimisation qui ne peut être résolue en un temps polynomial ou par une méthode exacte, sous la contrainte de caractéristiques de la fonction objectif (non convexité, continuité, dérivabilité...). Elle regroupe différentes typologies de problèmes :

— des problèmes à variables continues (non dénombrables), discrètes (dénombrables), ou mixtes ;
— des problèmes unimodaux ou multimodaux (possédant une ou plusieurs valeurs optimales) ;
— des problèmes mono, multi-objectifs (plusieurs objectifs souvent contradictoires), sous contraintes (faisabilité d'une solution) ;
— des problèmes d'optimisation statique ou dynamique, dans lesquels la fonction objectif, le domaine de définition peuvent restent identiques ou évoluer au cours du temps.

La résolution de cette classe de problèmes se fait alors par des méthodes approchées ou heuristiques, contenant généralement une composante aléatoire. Ces méthodes fournissent rapidement une valeur acceptable, sans garantie d'optimalité. Elles sont communément dédiées à un type de problème. Lorsqu'une heuristique est généralisable à plusieurs typologies de problèmes sans modification significative, on parle alors de métaheuristique.

Les métaheuristiques sont des algorithmes itératifs, possédant une composante aléatoire et parcourant l'espace de recherche par différentes techniques de génération de solutions. Ces algorithmes sont souvent inspirés par des systèmes physiques, biologiques ou éthologiques. Le caractère « méta » tient du fait qu'un même algorithme peut agréger différents problèmes d'optimisation difficile sans modification structurelle majeure. La partie dédiée au problème tient essentiellement en la représentation du problème et l'adaptation des opérateurs de recherche.

La figure 1.1 présente une fonction de coût à variable continue, définie sur le domaine de recherche $[0, 100]$. On considère un ensemble de six solutions formant une population. Le problème de minimisation de cette fonction possède une valeur optimale, *l'optimum global* x^*. Il possède également plusieurs *optima locaux* pouvant tromper la recherche. Les métaheuristiques vont itérativement déplacer ces solutions selon différentes stratégies afin de trouver une valeur approchée de x^*. Généralement une

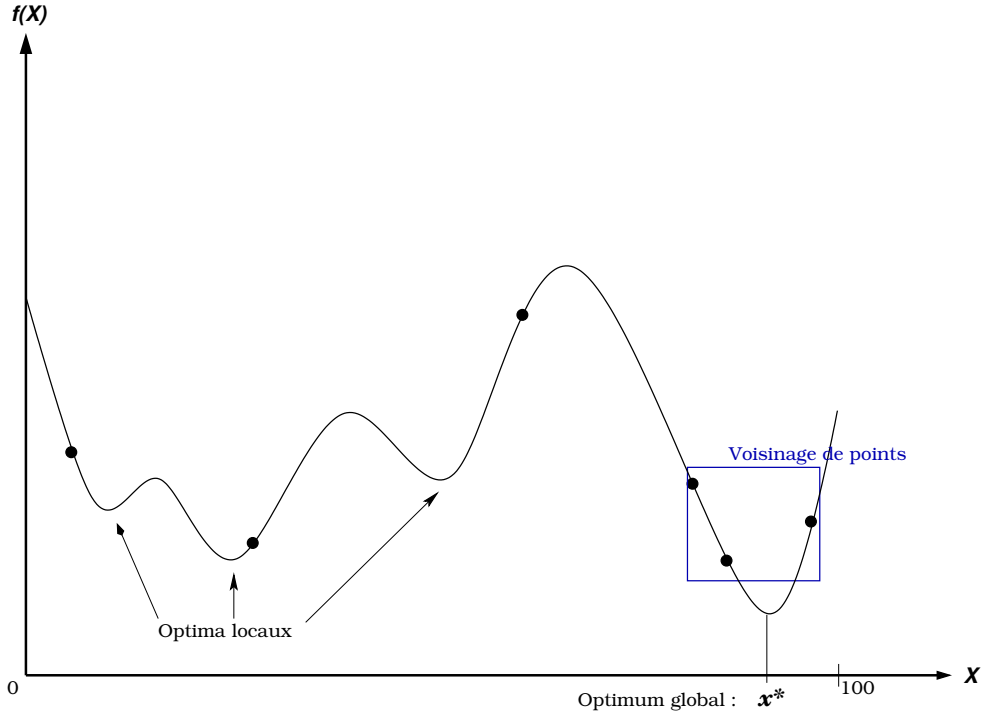

FIGURE 1.1 : Problème de minimisation en variables continues.

métaheuristique utilise des informations fournies par son voisinage propre (elle va se déplacer dans la direction qui améliore le résultat de la fonction), par un *voisinage* de solutions proches ou bien par d'autres bonnes solutions déjà trouvées.

Ces algorithmes réalisent le parcours du domaine de recherche selon deux comportements dichotomiques. Le premier est l'exploration, c'est-à-dire la capacité de l'algorithme à découvrir de nouvelles régions de l'espace de recherche. Ce comportement permet de ne pas être bloqué sur un optimum local mais ne favorise pas la convergence. Le second est l'exploitation : il s'agit de l'aptitude de la métaheuristique, utilisant une bonne solution, à continuer à chercher dans cette zone pour favoriser la convergence. Le risque est alors de provoquer une *convergence prématurée*, par exemple en attirant toutes les solutions vers un optimum local. Pour une métaheuristique, la difficulté est de réaliser un équilibre correct entre ces deux comportements afin de converger vers l'optimum global de l'espace de recherche, tout en évitant de rester bloqué sur une valeur d'optimum local.

Parmi les métaheuristiques d'optimisation globale les plus populaires, développées pour résoudre des problèmes d'optimisation discrète, on trouve des algorithmes de recherche locale, mono-agent (i.e. se basant sur une unique solution, modifiée itérativement), comme le recuit simulé (Kirkpatrick [Kirkpatrick et collab., 1983]) et la recherche Tabou (Glover [Glover, 1986]). Ces métaheuristiques ont la capacité de s'extraire d'une solution minimum locale et ainsi permettre de continuer à explorer le domaine de recherche de la fonction vers une meilleure solution.

Nous allons présenter un ensemble de métaheuristiques à base de population, appartenant à deux familles, les algorithmes évolutionnaires et l'intelligence en essaim, ainsi que leurs améliorations proposées dans plusieurs travaux de recherches.

1.2 Algorithmes évolutionnaires

Inspiré de la génétique et de la théorie de l'évolution, l'expression *algorithmes évolutionnaires* se décline en différentes sous-catégories, agrégeant des problématiques distinctes :

- la programmation évolutionnaire, issue des travaux de Fogel [Fogel et collab., 1966], vise à faire évoluer des structures d'automates finis par croisements et mutations successifs ;
- les algorithmes génétiques, métaheuristiques pour l'optimisation discrète, issus des travaux de Holland [Holland, 1975], où la solution est souvent représentée en nombre binaire ;
- la programmation génétique, développée par les travaux de Koza [Koza, 1989, 1990], est une méthode de création automatique de programmes, où les chromosomes sont des programmes informatiques ;
- l'évolution différentielle, une métaheuristique pour l'optimisation continue, de Price et Storm [Storn et Price, 1997] basée sur la mutation, le croisement et la sélection. Elle est abordée en 1.2.2 de ce chapitre ;
- les stratégies évolutionnaires, techniques d'optimisation continue, sont issues des travaux de Rechenberg [Rechenberg, 1989] et Schwefel [Schwefel, 1981], où une partie des solutions d'une population survit et génère de nouvelles solutions en utilisant le principe de mutation. Ces stratégies sont popularisées notamment par l'algorithme CMA-ES [Hansen et Ostermeier, 2001] de Hansen et Ostermeier.

Les termes utilisés en programmation évolutionnaire appartiennent au champ lexical de la génétique. Une solution à un problème est un *individu*, une expression du codage de la donnée est un *chromosome*, une sous-partie de ce codage, un *gène*. Un ensemble considéré de solutions est appelé une *population*. Les itérations faisant évoluer la population sont des *générations*. La population évolue grâce à un ensemble d'opérations : *sélection, croisement, mutation*. Les individus de la population impliqués dans une opération sont les *parents*, les individus résultats sont les *enfants* ou la *progéniture*.

Dans le cadre de l'optimisation difficile, nous allons présenter le principe des algorithmes génétiques ainsi qu'une étude détaillée de l'évolution différentielle.

1.2.1 Algorithme Génétique

Holland [Holland, 1975] puis Goldberg [Goldberg et Holland, 1988] développent l'algorithme génétique, une métaheuristique permettant à l'origine de résoudre des problèmes à variables discrètes.

C'est un algorithme élitiste à base d'une population de solutions. Cette population évolue durant plusieurs générations en sélectionnant, à chaque étape, les individus les plus performants. Certains individus se reproduisent, d'autres sont supprimés, un héritage génétique est transmis de génération en génération et conduit les individus les plus adaptés (i.e. répondant le mieux au problème) à survivre. Le processus est répété jusqu'à un certain critère d'arrêt.

1.2.1.1 Principe

Un algorithme génétique (Algorithme 1.1) consiste à appliquer itérativement un ensemble d'opérations définies comme suit.

Algorithme 1.1 : Algorithme génétique

```
   // Phase d'initialisation de la population, G = 0
1 : Initialisation des N individus selon une distribution uniforme
   // Phase d'évolution de la population
2 : tant que critère de fin non satisfait faire
       // Génération suivante G + 1
3 :    Sélection aléatoire de p individus parents
4 :    Croisement des p individus sélectionnés
5 :    Mutation des λ enfants obtenus
6 :    Évaluation des λ enfants obtenus
7 :    Sélection pour le remplacement
8 : fin
```

Sélection Elle permet de considérer un ensemble d'individus de la population sur la base de leurs performances. Généralement, on distingue deux opérateurs différents de sélection. La sélection parentale est destinée à la reproduction, tandis que la sélection de remplacement maintient la taille de la population constante, en choisissant les individus qui survivront lors de la prochaine génération. Une sélection retient de préférence les meilleurs individus mais doit aussi donner une chance aux moins bons pour éviter les problèmes de convergence prématurée. Cela peut se faire de plusieurs manières : aléatoirement, de manière élitiste, par le biais d'un tournoi, etc.

Croisement L'opération de croisement permet d'explorer l'espace de recherche en diversifiant la population. Elle manipule généralement les chromosomes de deux parents pour générer deux enfants. Il existe différents types de croisement, comme par exemple le croisement en un point (cf Figure 1.2), en deux points (cf Figure 1.3).

Mutation Cette opération reflète le caractère d'exploitation de l'algorithme. La mutation provoque une petite perturbation sur le chromosome d'un individu : un, ou plusieurs gènes, tirés aléatoirement, sont perturbés (cf Figure 1.4).

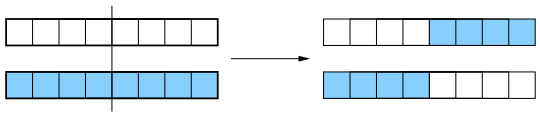

FIGURE 1.2 : Croisement en 1 point.

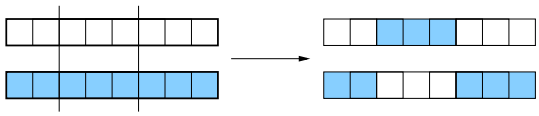

FIGURE 1.3 : Croisement en 2 points.

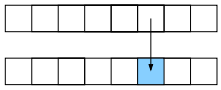

FIGURE 1.4 : Mutation.

Il existe de nombreuses adaptations pour le contexte continu, RCGA (Real-Coded Genetic Algorithm) [Chelouah et Siarry, 2000a; Davis, 1991; Michalewicz, 1996; Wright, 1991]. Par ailleurs, les stratégies évolutionnaires et l'évolution différentielle sont deux méthodes fondamentalement dédiées à l'optimisation en variables continues.

1.2.2 Évolution Différentielle

L'algorithme à évolution différentielle (*Differential Evolution*, DE) est inspiré des stratégies évolutionnaires et des algorithmes génétiques, et applicable à des problèmes à variables continues. Il a été proposé par Rainer Storn et Kenneth Price en 1997 [Storn et Price, 1997]. Il met en œuvre trois opérations issues des algorithmes évolutionnaires : la *mutation*, le *croisement* et la *sélection*. La méthode de génération d'un nouvel individu se fait en trois temps. En premier lieu, une mutation engendre un individu à partir d'un ensemble d'autres sélectionnés aléatoirement parmi la population. De multiples schémas de mutation ont été définis et seront détaillés par la suite. Il y a ensuite une phase de croisement, où différentes stratégies peuvent être employées pour générer un individu issu du croisement entre le parent et l'individu généré précédemment. La sélection par remplacement est alors effectuée.

1.2.2.1 Principe

L'évolution différentielle est un algorithme à population, constituée de N individus. Un individu $x_{i,G}$ est un vecteur de dimension D, où D est la dimension du problème et G représente la génération. Une population initiale $x_{i,0},\ i \in \{1,\ldots,N\}$ est créée par tirage aléatoire uniforme. L'algorithme effectue alors l'évolution de la population jusqu'à convergence. L'évolution se traduit itérativement par la création d'une nouvelle génération à partir des individus de la génération en cours, en appliquant les opérations de mutation et de croisement. La sélection s'effectue alors pour ne garder à la génération suivante que les meilleurs individus. Les figures 1.5 et 1.6, tirées de [Storn et Price, 1997], illustrent la création d'un nouvel individu (mutation et croisement).

Les différentes opérations impliquées dans la création d'une nouvelle génération sont les suivantes :

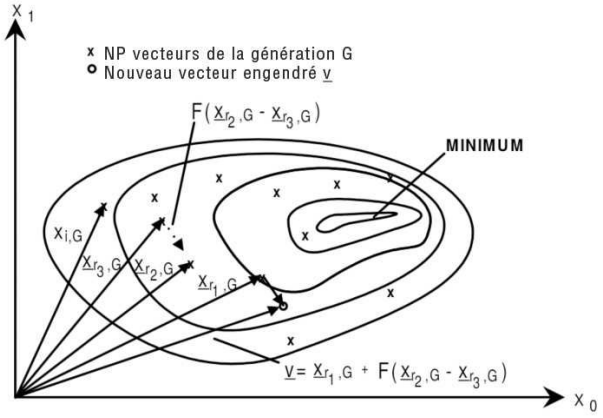

FIGURE 1.5 : Mutation

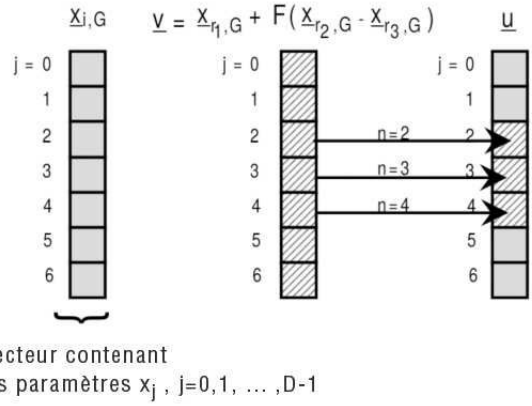

FIGURE 1.6 : Croisement exponentiel en 2 points.

Mutation Pour chaque individu cible $x_{i,G}$, un vecteur mutant $v_{i,G+1}$ est engendré selon :

$$v_{i,G+1} = x_{r1,G} + F\ (x_{r2,G} - x_{r3,G}). \tag{1.1}$$

où $r1,\ r2,\ r3 \in \{1, \ldots, N\}$ sont trois entiers différents entre eux et différents de i et $F \in [0, 2]$, une constante appelée *facteur d'amplification* contrôle l'amplitude de la différence $(x_{r2,G} - x_{r3,G})$.

Croisement Après la mutation, une opération de croisement entre les individus $v_{i,G+1}$ et $x_{i,G}$ est effectuée afin d'introduire de la diversité. Pour un opérateur de croisement binomial (*bin*), un vecteur $u_{i,G+1} = \{u_{1i,G+1}, \ldots, u_{Di,G+1}\}$ est généré par l'équation (1.2) :

$$u_{i,G+1} = \begin{cases} v_{ji,G+1} & \text{si } (rand_j(0,1) \leq CR) \text{ ou } (j = rnbr(i)), \\ & \forall\ j \in \{1, \ldots, D\}. \\ x_{ji,G} & \text{si } (rand_j(0,1) > CR) \text{ ou } (j \neq rnbr(i)). \end{cases} \tag{1.2}$$

Ici, $rand_j(0,1)$ est un nombre aléatoire $\in [0,1]$ tiré selon une loi de distribution uniforme pour la dimension j. $CR \in [0,1]$ est une constante appelée *taux de croisement* et $rnbr(i)$ est un indice de dimension choisi aléatoirement permettant de s'assurer qu'au moins un paramètre de l'individu mutant $v_{i,G+1}$ est transmis à $u_{i,G+1}$.

Pour un opérateur de croisement exponentiel (*exp*), on modifie une série consécutive (modulo la dimension D du problème) de L dimensions de la solution. On tire aléatoirement un indice initial n puis on détermine une longueur L de modification comme défini par l'algorithme 1.2.

Algorithme 1.2 : Longueur de croisement

```
1 : L ← 1
2 : tant que rand(0,1) < CR && L < D faire
3 : |  L ← L + 1
4 : fin
```

La figure 1.6 illustre un croisement exponentiel avec $n = 2$ et $L = 3$. Le vecteur $u_{i,G+1}$ est généré par l'équation (1.3).

$$u_{i,G+1} = \begin{cases} v_{ji,G+1} \quad (\text{de } n = rand(D) \text{ à } (n+L) \ mod \ D) \text{ ou } (j = rnbr(i)), \\ \\ x_{ji,G} \quad \text{sinon.} \end{cases} \quad \forall\, j \in \{1, \ldots, D\}. \quad (1.3)$$

Le croisement arithmétique, introduit dans [Price, 1999], modifie l'ensemble des composantes du vecteur parent selon (1.4), rendant obsolète l'utilisation du coefficient CR.

$$u_{i,G+1} = x_{ji,G} + rand(0,1)\ (v_{ji,G+1} - x_{ji,G}). \quad (1.4)$$

Une étude sur les effets du croisement sur l'algorithme a été réalisée par Zaharie [Zaharie, 2009]. Pour une même valeur de CR, la probabilité d'un fort taux de croisement est plus élevée dans le cas d'un croisement de type binomial. Cela conduit, dans le cas où le croisement exponentiel est appliqué, à choisir une valeur $CR > 0,9$ pour générer un croisement performant, alors qu'une valeur faible de CR suffit dans le cas du croisement binomial.

Sélection Afin de déterminer si l'individu généré appartiendra à la génération suivante $(G+1)$ de la population, un critère de sélection glouton est appliqué. Pour un problème de minimisation, on a :

$$x_{i,G+1} = \begin{cases} u_{i,G+1} \quad \text{si } f(u_{i,G+1}) < f(x_{i,G}), \\ \\ x_{i,G} \quad \text{sinon.} \end{cases} \quad (1.5)$$

Cet algorithme possède donc un nombre restreint de paramètres : la taille de la population N ainsi que les coefficients F et CR.

Le processus général est décrit par l'algorithme 1.3.

La taille de la population N permet, lorsque sa valeur est élevée, de davantage explorer l'espace de recherche et augmente la diversité de directions possibles (i.e. les différences entre paires d'individus). Néanmoins, une valeur élevée diminue la capacité de convergence de l'algorithme. Le paramètre CR contrôle l'influence de l'individu cible, F influe sur l'importance de la trajectoire (i.e. la différence calculée en équation (1.1)).

Algorithme 1.3 : Algorithme DE/rand/1/bin

```
// Phase d'initialisation de la population, G = 0
Initialisation des N individus selon une distribution uniforme
// Phase d'évolution de la population
tant que critère de fin non satisfait faire
    // Création de la (G+1)^ieme génération
    pour i ← 1 à N faire
        Sélection aléatoire de trois individus distincts de x_i et distincts en entre eux :
        x_{r1,G}, x_{r2,G}, x_{r3,G}
        // Mutation
        Génération de l'individu v_{i,G+1}, selon (1.1)
        // Croisement
        Création de l'individu u_{i,G+1}, selon (1.2)
    fin
    // Sélection
    pour i ← 1 à N faire
        x_{i,G+1} devient le meilleur individu entre u_{i,G+1} et x_{i,G}, selon (1.5)
    fin
fin
```

1.2.2.2 Améliorations

Bien que l'algorithme de base possède peu de paramètres, il est très sensible au choix du facteur d'amplification et du taux de croisement, contrôlant ses caractères d'exploration et de diversité. Au-delà de l'apparente simplicité de configuration, beaucoup de travaux de recherche portent sur l'auto-adaptation des différents paramètres. De nombreuses stratégies de mutation et de croisement ont ainsi été proposées dans la littérature et offrent de nouvelles possibilités de combinaisons et d'adaptations.

Les paramètres Une étude sur l'influence des paramètres [Gämperle et collab., 2002] propose de choisir un nombre d'individus N entre 3 et 8 fois la dimension. Les auteurs recommandent de ne pas choisir un facteur amplification F trop petit, pour éviter une convergence prématurée, ni trop grand ($F > 1$), car les capacités de convergence décroissent. La valeur conseillée est $0,6$. Enfin, pour la constante du taux d'accroissement CR, ils déconseillent de prendre des valeurs au-delà de $0,9$. Un intervalle $[0,3, 0,9]$ est proposé. Une valeur faible ralentit la convergence (car peu de composantes de l'individu cible sont modifiées). Dans [Montgomery, 2010], Montgomery réalise une étude approfondie du comportement de l'algorithme selon ces trois paramètres, en poussant son étude particulièrement sur le paramètre CR.

Les schémas de mutation Le schéma de mutation présenté en équation (1.1) est un des schémas proposés par les auteurs. Il existe plusieurs variations de l'équation de mutation, décrites ci-après, basées sur le modèle suivant : $DE/x/y/z$ où :

— x désigne le mode de sélection de l'individu auquel on applique la mutation. Dans l'équation (1.1), il est choisi aléatoirement, la méthode est nommée *rand*, ou par exemple *best* pour le meilleur individu de la population ;
— y désigne le nombre de différences utilisées ;
— z désigne le schéma de croisement, la variante présentée ici est *bin* pour binomiale (mais aussi *exp* pour exponentielle).

Le schéma de la méthode d'évolution par défaut est alors $DE/rand/1/bin$. Une liste non exhaustive des mutations les plus rencontrées dans la littérature est présentée :

— **DE/rand/2/bin** : une deuxième différence est ajoutée au calcul, cinq individus distincts sont nécessaires.

$$v_{i,G+1} = x_{r1,G} + F\ (x_{r2,G} - x_{r3,G}) + F\ (x_{r4,G} - x_{r5,G}). \tag{1.6}$$

— **DE/best/1** : l'individu auquel est appliquée la mutation est le meilleur de la population.

$$v_{i,G+1} = x_{best,G} + F\ (x_{r1,G} - x_{r2,G}). \tag{1.7}$$

— **DE/best/2** : identique au précédent, en ajoutant une différence entre deux individus distincts supplémentaires.

$$v_{i,G+1} = x_{best,G} + F\ (x_{r1,G} - x_{r2,G}) + F\ (x_{r3,G} - x_{r4,G}). \tag{1.8}$$

— **DE/rand-to-best/2** : on ajoute ici à une solution tirée aléatoirement la différence correspondant à la direction entre l'individu cible et le meilleur individu de la population.

$$v_{i,G+1} = x_{r1,G} + F\ (x_{r2,G} - x_{r3,G}) + F\ (x_{r4,G} - x_{r5,G}) + F\ (x_{best,G} - x_{i,G}). \tag{1.9}$$

— **DE/current-to-best/1** (ou *DE/target-to-best/1*) : identique au précédent, à ceci près que l'individu auquel est appliquée la mutation est l'individu cible, et l'on ajoute la différence correspondant à la direction entre ce dernier et le meilleur individu de la population.

$$v_{i,G+1} = x_{i,G} + F\ (x_{r1,G} - x_{r2,G}) + F\ (x_{best,G} - x_{i,G}). \tag{1.10}$$

— **DE/current-to-rand/1** : on fait intervenir la direction entre l'individu cible et un individu supplémentaire, distinct.

$$v_{i,G+1} = x_{i,G} + K\ (x_{r3,G} - x_{i,G}) + F'\ (x_{r1,G} - x_{r2,G}). \tag{1.11}$$

Le coefficient K est le coefficient de combinaison, choisi aléatoirement dans $[0, 1]$ et $F' = K\ F$. Dans ce schéma, il n'y a pas de croisement après mutation.

— **DE/rand/1/either-or** : le schéma de génération est sensiblement différent :

$$v_{i,G+1} = \begin{cases} x_{r1,G} + F\ (x_{r2,G} - x_{r3,G}) & \text{si } U(0,1) < P_F, \\ \\ x_{r3,G} + K\ (x_{r1,G} + x_{r2,G} - 2x_{r3,G}) & \text{sinon.} \end{cases} \tag{1.12}$$

où le paramètre $K = 0,5\ (F+1)$, on note que $0,4$ est conseillé comme valeur pour P_F. Comme en équation (1.11), il n'y a pas de croisement après mutation.

Multi-stratégies/population Face aux diversités des schémas de mutation, des différents modes de croisement et des valeurs possibles pour les paramètres F et CR, des stratégies hybrides comportant un ensemble de comportements ou plusieurs populations ont été développées.

Mallipeddi et Suganthan [Mallipeddi et collab., 2011] proposent dans leur variante EPSDE, un ensemble de schémas de mutation composé de {DE/current-to-rand/1 (1.11) pour l'exploration, JADE (1.22) pour l'exploitation}, un ensemble de croisements {bin, exp} et deux ensembles de valeurs pour F et pour CR. À l'initialisation, ils affectent aléatoirement à chaque individu chacune de ces caractéristiques, parmi l'ensemble des possibilités. Si un individu généré est meilleur que son parent, il hérite des caractéristiques. Si ce n'est pas le cas, de nouvelles caractéristiques lui sont réaffectées aléatoirement.

Dans le même esprit, Wang *et al.* [Wang et collab., 2011] utilisent un pool de mutations {*DE/rand/-1/bin* (1.1), *DE/rand/2/bin* (1.6), *DE/current-to-rand/1* (1.11)} ainsi qu'un ensemble de couples (F, CR). Ici, les compositions entre mutation et coefficient sont sélectionnées aléatoirement et non pas selon un apprentissage de réussite. Trois candidats sont également générés pour chaque parent. Pour le schéma *DE/current-to-rand/1*, le croisement arithmétique (1.4), invariant aux rotations est préféré au croisement binomial.

Une autre stratégie est d'utiliser plusieurs sous-populations, chacune appliquant une stratégie différente. Dans [Lynn et collab., 2015], les auteurs proposent d'attribuer un comportement exploratoire à une première population et d'utiliser une seconde population, pour l'exploitation des zones les plus prometteuses. À la première sous-population sont affectés aléatoirement deux schémas de mutation à caractère exploratoire : *DE/current-to-rand/1* (1.11) et *DE/rand/2* (1.6). La seconde sous-population, dédiée à l'exploitation, est basée sur le schéma *DE/current-to-pbest/1* (1.22). Les p% meilleurs individus sont sélectionnés sur l'ensemble de la population, et une archive est utilisée de surcroit (voir la description de JADE ci-après).

Inspiré de PSO, cf section 1.3.2, et du schéma de mutation *DE/target-to-best/1* (1.10), Das *et al.*, dans leur algorithme DEGL [Das et collab., 2009], prennent en compte le meilleur individu global de la population (éq. (1.13)) et le meilleur individu dans un voisinage (éq. (1.14)) de l'individu parent considéré.

$$glob_{i,G+1} = x_{i,G} + \alpha\ (x^p_{gbest,G} - x_{i,G}) + \beta\ (x_{r_1,G} - x_{r_2,G}). \tag{1.13}$$

$$loc_{i,G+1} = x_{i,G} + \alpha\ (x^p_{lbest,G} - x_{i,G}) + \beta\ (x_{p,G} - x_{q,G}). \tag{1.14}$$

où α, β sont deux coefficients d'amplification (ils considèrent $\alpha = \beta = F$), p et q deux indices différents choisis dans le voisinage de la i^{ieme} solution considérée. La solution candidate est générée par une pondération des déplacements (1.15) vers les solutions globales et locales. Le poids w est un paramètre de l'algorithme. La version de base a $w = 0,5$ afin de ne favoriser aucun des deux comportements.

$$v_{i,G+1} = w\ glob_{i,G+1} + (1 - w)\ loc_{i,G+1}. \tag{1.15}$$

Ils proposent alors différentes méthodes d'adaptation du poids w (auto-incrément, aléatoire, auto-adaptation), l'auto-adaptation se révélant la méthode la plus performante.

Dans [Rahnamayan et collab., 2008], les auteurs présentent une technique s'appuyant sur l'apprentissage basé sur l'opposition. Fondé sur le principe du nombre opposé : *soit* $x \in [a, b]$, $\breve{x} = a + b - x$, leur algorithme modifie l'initialisation et le déplacement. Pour l'initialisation, il génère $2N$ individus (N et leurs opposés), et sélectionne les N meilleurs. Pour le déplacement, l'algorithme réalise un déplacement classique de population, puis selon une probabilité de saut (Jr), les points opposés sont calculés selon : $\breve{x}_{ij,G} = x_{min_j} + x_{max_j} - x_{ij,G}$, où x_{min_j} et x_{max_j} sont les bornes de la j^{ieme} dimension. Comme pour l'initialisation, les N meilleurs sont retenus.

Wu *et al.* [Wu et collab., 2016] divisent la population en quatre. Aux trois premières parties, dites *indicatrices*, de taille égale, sont affectés trois schémas de mutation parmi {*DE/current-to-pbest/1* (1.22), *DE/rand/1* (1.1), *DE/current-to-rand/1* (1.11)}. À la dernière sous-population, on attribue aléatoirement un des trois schémas. Cette population, dite de *récompense*, sert à donner à l'algorithme plus de ressources de calcul pour un schéma de mutation efficace. En effet, après un certain nombre de générations (un paramètre), l'algorithme évalue les performances de chaque sous-population puis affecte le schéma le plus performant à la dernière population. La taille de la dernière population est supérieure à chacune des trois premières. Les auteurs proposent que chacune des populations *indicatrices* soit de taille $20\%N$ et la population de *récompense* soit $40\%N$. Les sous-populations sont reconstituées à chaque génération. Les coefficients F et CR sont adaptés selon le modèle de JADE (cf ci-après).

L'auto-adaptation Les performances de l'algorithme sont très sensibles au choix initial des valeurs du facteur d'amplification et du taux de croisement. Aussi, ces coefficients devraient pouvoir évoluer au cours du temps pour éviter une convergence prématurée et revenir vers un comportement exploratoire si nécessaire. Plusieurs travaux permettant de faire évoluer les valeurs de F et CR ont été développés.

Parmi les plus populaires, Brest *et al.* proposent l'algorithme jDE [Brest et collab., 2006], dans lequel, à chaque individu i, sont associés des coefficients distincts F_i et CR_i. L'algorithme fait évoluer ces coefficients à chaque génération selon (1.16) et (1.17).

$$F_{i,G+1} = \begin{cases} F_{min} + rand(0,1)\ F_{max} & \text{si } rand(0,1) < \tau_1, \\ \\ F_{i,G} & \text{sinon.} \end{cases} \tag{1.16}$$

$$CR_{i,G+1} = \begin{cases} rand(0,1) & \text{si } rand(0,1) < \tau_2, \\ \\ CR_{i,G} & \text{sinon.} \end{cases} \tag{1.17}$$

L'algorithme introduit donc les paramètres $\tau_1 = \tau_2 = 0,1$, $F_{min} = 0,1$, $F_{max} = 0,9$.

L'algorithme SaDE, développé par Quin *et al.* [Qin et collab., 2009], effectue non seulement une adaptation des coefficients F et CR mais aussi de la stratégie de mutation. Une valeur de F_i, tirée aléatoirement à chaque génération, est associée à un individu et une mémoire de coefficient CR est créée pour chaque stratégie de mutation impliquée dans l'algorithme. L'adaptation de stratégie est réalisée selon un apprentissage des expériences passées. La stratégie de mutation est choisie selon une probabilité correspondant à un taux de succès passé. La taille de la population est laissée en paramètre.

Zhang *et al.* dans [Zhang et Sanderson, 2009], travaillent eux aussi sur l'adaptation des coefficients F et CR. Dans l'algorithme JADE, ces coefficients sont associés à chaque individu et une mémoire des coefficients des meilleurs individus est utilisée. L'adaptation se fait en utilisant des tirages aléatoires suivant une loi normale pour CR et pour F, une loi de Cauchy de moyenne, la moyenne du coefficient dans la mémoire, et d'écart-type $0,1$ (équations (1.18) et (1.19)).

$$CR_i = rand_{norm}\ (\mu_{CR},\ 0,1). \tag{1.18}$$

$$F_i = rand_{cauchy}\ (\mu_F,\ 0,1). \tag{1.19}$$

où :

$$\mu_{CR} = (1-c)\ \mu_{CR} + c\ moy(S_{CR}). \tag{1.20}$$

$$\mu_F = (1 - c)\ \mu_F + c\ moy_L(S_F). \tag{1.21}$$

Les paramètres d'adaptation μ_{CR} et μ_F sont initialisés à 0,5, c est le taux d'apprentissage, dont la valeur conseillée est $0,1$, S_{CR} et S_F sont les sauvegardes des coefficients ayant conduit à une meilleure solution pour chaque individu et moy_L est la moyenne de Lehmer, donnée par :

$$moy_L(S_F) = \frac{\sum_{F \in S_F} F^2}{\sum_{F \in S_F} F}.$$

Les auteurs introduisent aussi un nouveau schéma de mutation : *DE/current-to-pbest/1* (1.22), une généralisation du schéma *DE/current-to-best/1*, où p représente un pourcentage de bonnes solutions visitées à prendre en compte.

$$v_{i,G+1} = x_{i,G} + F_i\ (x_{r_1,G} - x_{r_2,G}) + F_i\ (x^p_{best,G} - x_{i,G}). \tag{1.22}$$

où $x^p_{best,G}$, est un vecteur choisi aléatoirement parmi les 100p% meilleurs individus de la population. Ce schéma de mutation est repris dans de nombreux papiers [Lynn et collab., 2015; Tanabe et Fukunaga, 2013; Wu et collab., 2016]. Enfin, une archive externe peut être utilisée, permettant de stocker des solutions parentes abandonnées ; le cas échéant, $x_{r_2,G}$ est choisi aléatoirement parmi la population courante et l'archive.

Tanabe et Fukunaga proposent SHADE [Tanabe et Fukunaga, 2013], une version améliorée de l'algorithme précédent JADE. Se basant sur le même principe que JADE, ils introduisent une mémoire étendue pour sauvegarder un historique de taille H des moyennes des coefficients S_{CR} et S_F. Les équations (1.18) et (1.19)) deviennent :

$$CR_i = rand_{norm}(M_{CR,r_i},\ 0,1). \tag{1.23}$$

$$F_i = rand_{cauchy}(M_{F,r_i},\ 0,1). \tag{1.24}$$

où $r_i \in [1, H]$, M_{CR,r_i} et M_{F,r_i} sont respectivement les moyennes pondérées des coefficients S_{CR} et S_F, initialisées à $0,5$.

Ils suggèrent également d'affecter le coefficient p, du schéma de mutation *DE/current-to-pbest/1*, à chaque individu. Ce coefficient est recalculé à chaque génération selon $p_i = rand(2/N,\ 0,2)$.

L-SHADE [Tanabe et Fukunaga, 2014], une amélioration de l'algorithme SHADE, introduit une réduction progressive de la population selon une fonction linéaire. Une taille initiale de population N^{init} et une taille minimale N^{min} sont définies. À chaque génération, la taille de la population est

recalculée selon (1.25)

$$N^{G+1} = round\left[\left(\frac{N^{min} - N^{init}}{max_{eval}}\right) nb_{eval} + N^{init}\right]. \tag{1.25}$$

La valeur de N^{min} dépend du schéma de mutation utilisé, ici *DE/current-to-pbest/1* nécessite au moins quatre individus. nb_{eval} est le nombre d'évaluations en cours et max_{eval} est le nombre maximum d'évaluations. Les $N^{G+1} - N^{G}$ plus mauvais individus sont supprimés de la population.

Une autre stratégie adaptative proposée par Tang *et al.* [Tang et collab., 2015] consiste à faire dépendre les valeurs F et CR de la valeur de la fonction objectif pour l'individu considéré. Plus un individu est proche de l'optimum, plus il doit transmettre de caractéristiques à son descendant et moins il doit être déplacé (valeurs de CR et F petites). Le calcul des coefficients est basé sur la même formule, F_i ou $CR_i = \frac{f_i - f_L + \delta_L}{f_U - f_L + \delta_L}$, où f_L et f_U sont respectivement les valeurs minimum et maximum de la fonction objectif pour la population courante, δ_L est la valeur absolue de la différence entre les deux plus petites valeurs distinctes de la fonction objectif trouvées jusqu'alors et f_i la valeur de la fonction objectif pour l'individu considéré. La valeur du coefficient considéré est tirée aléatoirement avec un écart-type de $0,1$ ($F = rand(F_i,\ 0,1)$).

En 2001, une bibliographie complète et un ensemble d'applications ont été réalisés par J. Lampinen [Lampinen, 2001] ; de plus récents états de l'art peuvent être consultés dans [Das et Suganthan, 2011; Neri et Tirronen, 2010; S. Das, 2016].

1.3 L'intelligence en essaim

L'intelligence en essaim désigne le comportement d'insectes sociaux dont les interactions donnent une cohérence au groupe. Malgré leur éloignement du milieu biologique, le terme a été introduit par Beni et Wang [Beni et Wang, 1993], pour décrire un ensemble de robots travaillant en coopération afin de résoudre un problème. Bonabeau *et al.* [Bonabeau et collab., 1999] caractérisent l'intelligence en essaim par un ensemble d'individus simples, communicants et formant une population auto-organisée, adaptative et capable de résoudre des problèmes complexes.

Dans le cadre de métaphores biologiques et éthologiques, un essaim correspond à une volée d'oiseaux, une colonie de fourmis ou d'abeilles, une population de bactéries, etc. La thèse de Rodolphe Charrier [Charrier, 2009] offre une bibliographie détaillée sur l'intelligence en essaim.

Outre les métaheuristiques présentées dans cette section, d'autres méthodes basées sur le concept d'intelligence en essaim ont été proposées, nous pouvons citer :

— *BFO* [Passino, 2002] (pour *Bacterial Foraging Optimisation*), s'inspirant du comportement de prolifération de la bactérie E-Coli (*Escherichia Coli*) ;

— la recherche coucou [Yang et Deb, 2009] (*Cuckoo Search*), propose une exploration basée sur le comportement parasitique et la reproduction du coucou, couplée à une recherche locale basée sur les sauts de Lévy [Mandelbrot, 1977] ;

— l'algorithme des lucioles [Yang, 2010] (*Firefly Algorithm*), dans lequel les lucioles communiquent par signaux lumineux.

1.3.1 Algorithme de Colonies de Fourmis

Le terme colonies de fourmis est un terme générique représentant une classe d'algorithmes, initiée par l'algorithme « Système de Fourmis » (Ant System) de Colorni, Dorigo et Maniezzo [Colorni et collab., 1992].

Cet algorithme se base sur le comportement de communication particulier des fourmis, la *stigmergie*. Lorsqu'elles explorent un environnement, les fourmis construisent un chemin en déposant une substance volatile, la phéromone. Les suivantes « lisent » cette information grâce à leurs antennes et choisiront probablement le chemin possédant la plus forte concentration de phéromone.

La figure 1.7 illustre ce comportement lors d'une expérience menée par Goss *et al.* [Goss et collab., 1989] sur le comportement auto-organisationnel des fourmis argentines. Dans un premier temps, une fourmi sélectionne un chemin a de son nid N jusqu'à la source de nourriture F, puis revient au nid par le chemin b en déposant une quantité de phéromone. Ensuite, chaque fourmi de la colonie emprunte les différents chemins et en retournant à N, elles renforcent en phéromone la piste la plus courte, la rendant plus attractive. Ce comportement, ainsi que le phénomène d'évaporation de phéromone, vont écarter les chemins les plus longs, la colonie aura choisi le chemin le plus court.

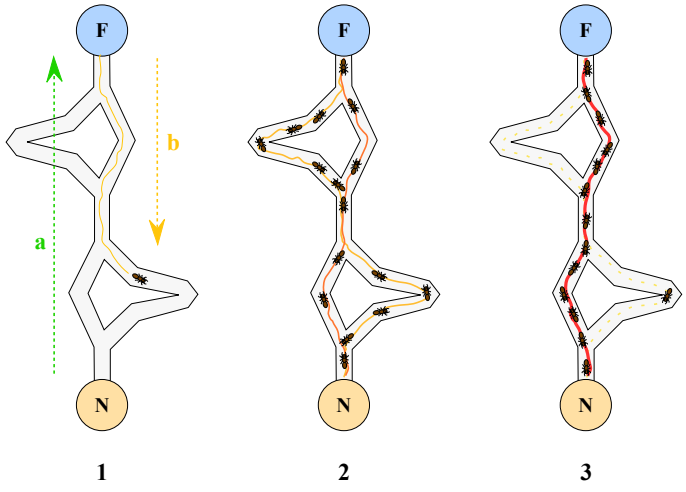

FIGURE 1.7 : Expérience de sélection du chemin le plus court par une colonie de fourmis (J. Dréo).

Créé pour répondre au problème du voyageur de commerce (*Traveling Salesman Problem* (TSP), abordé en introduction générale), l'algorithme de système de fourmis est la première métaheuristique

basée sur une métaphore d'individus sociaux à mémoire centralisée, la matrice de quantités de phéromone.

1.3.1.1 Principe

Cette métaheuristique repose sur le phénomène de dépôt d'une substance chimique volatile, la phéromone, par une fourmi permettant de communiquer avec les autres membres de la colonie. Soient n villes à relier et une colonie composée de m fourmis. À chaque itération t, chaque fourmi k construit une tournée en reliant une ville i à une autre ville j, pas encore visitée. Pour cela, on considère :

— une liste de villes à visiter par la fourmi k, lorsqu'elle est en i : J_i^k ;

— une variable représentant la visibilité d'une ville j depuis une ville i, dépendant de la distance les séparant : $\eta_{ij} = \frac{1}{d_{ij}}$;

— une quantité de phéromone entre deux villes, représentant l'intensité de l'utilisation de ce trajet : τ_{ij} ;

— trois paramètres α, β et Q.

Le déplacement de la fourmi est défini par l'équation (1.26), où $p_{ij}^k(t)$ représente la probabilité que la fourmi k, placée en i, choisisse la ville j. La visibilité, η_{ij}, aide la fourmi à choisir une ville proche, l'intensité de la phéromone, τ_{ij}, permet à la fourmi de choisir un chemin possédant une grande quantité de phéromone. Les paramètres de l'algorithme α et β équilibrent l'importance de ces deux variables.

$$p_{ij}^k(t) \;=\; \begin{cases} \dfrac{\tau_{ij}^\alpha(t)\ \eta_{ij}^\beta}{\displaystyle\sum_{l \in J_i^k} \tau_{il}^\alpha(t)\eta_{il}^\beta} & \text{si } j \in J_i^k, \\ 0 & \text{sinon.} \end{cases} \tag{1.26}$$

Après avoir effectué le chemin, la fourmi dépose une quantité de phéromone, $\Delta\tau_{ij}^k(t)$, dépendant de la qualité de la solution (i.e. la longueur totale du chemin trouvé, $L^k(t)$). Elle est donnée par :

$$\Delta\tau_{ij}^k(t) \;=\; \begin{cases} \frac{Q}{L^k(t)} & \text{si } (i,j) \in T^k(t), \\ 0 & \text{sinon.} \end{cases} \tag{1.27}$$

où $T^k(t)$ est le trajet déjà effectué par la fourmi k à l'itération t et Q est une constante de l'algorithme. Donc plus un chemin construit est long, moins la quantité de phéromone déposée sera importante.

Enfin, avant le cycle suivant, on effectue une évaporation de la quantité de phéromone sur chacune des pistes. Le but ici est d'oublier (de ne pas prendre en compte) les mauvaises solutions. Cette décroissance constante est calculée selon :

$$\tau_{ij}(t+1) = (1-\rho)\ \tau_{ij}(t) \;+\; \sum_{k=1}^{m} \Delta\tau_{ij}^k(t). \tag{1.28}$$

où m est le nombre de fourmis et ρ, le taux d'évaporation, un paramètre de l'algorithme.

L'algorithme 1.4 résume le processus. $T^k(t)$ est le trajet complet réalisé par la k^{ieme} fourmi.

Algorithme 1.4 : Algorithme de Système de Fourmis

```
  // Phase d'initialisation
1 : t ← 0
2 : τ_ij(t) ← τ_0,  ∀i, j ∈ {1, ..., n}
  // Phase de déplacement de la colonie
3 : tant que critère de fin non satisfait faire
      // Pour chaque fourmi
4 :   pour k = 1 à m faire
        // Construction d'un trajet T^k
5 :     Choisir aléatoirement une ville
6 :     pour chaque ville non visitée i faire
7 :       Choisir une ville j parmi J_i^k villes restantes selon (1.26)
8 :     fin
9 :     Dépôt d'une quantité de phéromone Δτ_ij^k(t), sur le trajet T^k(t), selon (1.27)
10 :  fin
11 :  Évaporation des phéromones selon (1.28)
12 :  t ← t + 1
13 : fin
```

L'algorithme de « Système de Colonie de Fourmis » (Ant Colony System) proposé par Dorigo *et al.* [Dorigo et Gambardella, 1997], illustré par l'algorithme 1.5, est une amélioration du système de fourmis. Il inclut, lors de la construction d'un trajet, deux techniques de liaison de deux villes favorisant, selon une probabilité q_0, l'exploitation ou l'exploration.

$$j \in J_i^k, \text{ est choisie selon } \begin{cases} j = \arg max_{l \in J_i^k} \left\{ \tau_{il}(t)\ \eta_{il}^{\beta} \right\} & \text{si } rand(0,1) \leq q_0, \\ \text{la probabilité } p_{ij}^k(t) = \dfrac{\tau_{ij}(t)\ \eta_{ij}^{\beta}}{\sum\limits_{l \in J_i^k} \tau_{il}(t) \eta_{il}^{\beta}} & \text{sinon.} \end{cases} \tag{1.29}$$

Nous remarquons qu'avec la probabilité q_0, la fourmi choisira la ville proche, selon un critère d'exploitation. En effet, la quantité représente la maximisation d'une variable dont l'expression est l'inverse de la distance. Aussi, lorsqu'un trajet est construit par une fourmi, les auteurs adjoignent une technique de recherche locale afin de l'améliorer. Enfin, seul le meilleur trajet est retenu et la mise à jour de la quantité de phéromone (cf éq. (1.30)) ne se fait que sur les arcs appartenant à ce chemin.

$$\tau_{ij}(t+1) = (1-\rho)\ \tau_{ij}(t)\ +\ \rho\ \frac{1}{L^+},\ \forall (i,j) \in T^+. \tag{1.30}$$

où T^+ est le meilleur trajet trouvé lors d'un cycle et L^+, sa longueur. L'évaporation (cf éq. (1.31)) est réalisée localement à chaque passage de fourmi sur un arc reliant deux villes, lors de la construction

du trajet.

$$\tau_{ij}(t) = (1 - \epsilon)\ \tau_{ij}(t)\ +\ \epsilon\ \tau_0. \tag{1.31}$$

où ϵ est le pourcentage d'évaporation et τ_0, la quantité initiale de phéromone. À noter ici que l'évaporation de phéromone (tendant vers τ_0) se fera de manière plus importante à mesure qu'un trajet sera utilisé. Cela permet à l'algorithme de ne pas converger trop rapidement en forçant les fourmis à découvrir de nouveaux chemins.

Algorithme 1.5 : Algorithme de Système de Colonie de Fourmis

```
   // Phase d'initialisation
1 : t ← 0
2 : τ_ij(t) ← τ_0,  ∀i, j ∈ {1, ..., n}
   // Phase de déplacement de la colonie
3 : tant que critère de fin non satisfait faire
       // Pour chaque fourmi
4 :    pour k = 1 à m faire
5 :       Construction d'un trajet T^k en évaporant les phéromones selon (1.29) et (1.31)
6 :       Calcul du coût L^k du trajet T^k
7 :       Appliquer une recherche locale sur T^k pour l'améliorer
8 :    fin
9 :    Soit T^+ le meilleur trajet trouvé par les m fourmis
10 :   Mise à jour des phéromones sur les arcs de T^+ selon (1.30)
11 :   t ← t + 1
12 : fin
```

1.3.1.2 Adaptation au contexte continu

Tout comme les premières métaheuristiques telles que le recuit simulé, la recherche tabou ou les algorithmes génétiques, les problématiques agrégées par les ACO sont à variables discrètes. En effet, les algorithmes de colonie de fourmis ont été d'abord dédiés au TSP puis diversifiés à d'autres problèmes combinatoires [Dorigo et Stützle, 2003]. Ils gardent en commun la construction itérative d'une solution ainsi que leur caractère discret (nombre fini de valeurs possibles).

Plusieurs adaptations aux problèmes à variables continues ont été proposées. Nous pouvons citer l'algorithme CACO proposé par Bilchev et Parmee [Bilchev et Parmee, 1995], dont le principe, assez éloigné de l'algorithme original, introduit le concept de nid et de vecteur de directions. Une direction est choisie au hasard par une fourmi. Elle se déplace selon cette direction et réalise ensuite une recherche locale. Une recherche globale, basée sur un algorithme génétique, est réalisée pour mettre à jour le vecteur de directions.

L'algorithme API [Monmarché et collab., 2000] est également éloigné de l'ACO en version discrète, n'utilisant pas la communication par phéromone. Dans cette méthode, les fourmis sont envoyées depuis

un nid vers différentes zones de l'espace de recherche puis effectuent plusieurs recherches locales et gardent la meilleure. Les fourmis comparent leurs performances deux à deux, la fourmi la moins performante est envoyée vers la meilleure. Périodiquement, le nid est recentré vers la meilleure zone visitée.

Avec CIAC [Dréo et Siarry, 2002] (*Continuous Interacting Ant Colony*), Dréo et Siarry changent le paradigme et introduisent le concept de canal de communication. Les fourmis communiquent selon deux types de canaux : stigmergique et direct. Par le premier, une fourmi se déplace vers le centre de gravité des zones d'intérêt visitées. Par le second, les fourmis disposent d'une pile de messages reçus. Elles en lisent aléatoirement un et se déplacent vers l'expéditeur si sa zone est plus prometteuse. L'algorithme HCIAC [Dréo et Siarry, 2007] propose une amélioration par hybridation d'une recherche locale de Nelder-Mead [Nelder et Mead, 1965].

Socha et Dorigo développent $ACO_{\mathbb{R}}$ [Socha et Dorigo, 2008], une version dans laquelle le vecteur solution est construit itérativement, selon le modèle du système de fourmis, grâce à une fonction améliorée de densité de probabilité gaussienne associée à chaque dimension du problème. La fonction est élaborée par le biais d'une archive ordonnée de solutions, de leur évaluation et de leur poids. La mise à jour des phéromones consiste à remplacer les moins bonnes solutions de l'archive par de nouvelles. Cet algorithme a inspiré des travaux récents, proposant des améliorations $DACO_{\mathbb{R}}$ [Leguizamón et Coello, 2010], $IACO_{\mathbb{R}} - LS$ [Liao et collab., 2011], et une version unifiée $UACO_{\mathbb{R}}$ [Liao et collab., 2014] de Liao *et al.*

Les tentatives d'adaptation des ACO au contexte continu obligent les auteurs à réaliser des compromis avec la définition de base. Chacune redéfinit la métaphore de base d'une manière propre, ne gardant que peu de points communs les unes avec les autres. Nous allons maintenant présenter des métaheuristiques de la famille de l'intelligence en essaim, dédiées à la résolution de problèmes en variables continues.

1.3.2 Algorithme d'Optimisation par Essaim Particulaire

L'algorithme « d'Optimisation par Essaim Particulaire » ou *Particle Swarm Optimization* (PSO) est un algorithme à population, proposé en 1995 par Russel Eberhart et James Kennedy [Kennedy et Eberhart, 1995]. Contrairement aux algorithmes basés sur des opérateurs génétiques, il ne se base pas sur la sélection, le croisement des meilleurs individus mais sur la collaboration entre eux. De plus, contrairement aux ACO, il ne possède pas de mémoire centralisée. La population est ainsi constituée d'un ensemble d'agents, ou particules, peu intelligents (i.e. qui n'ont qu'une connaissance restreinte de leur milieu) mais fortement communicants (selon une architecture définie de voisinage). L'analogie animalière correspond aux bancs de poissons ou aux vols d'étourneaux possédant une dynamique de

déplacement complexe. Cette dynamique respecte des contraintes implicites du groupe, de direction et de vitesse permettant de construire une organisation cohérente.

1.3.2.1 Principe

Chaque particule représente une solution du problème. Elle possède une position (une solution du problème) et une « vitesse ». La vitesse est en fait un vecteur de déplacement, de modification de sa position actuelle, définissant sa future position probable. De plus, chaque particule possède une mémoire de la meilleure position visitée jusqu'alors ainsi que de la meilleure position de son groupe d'informatrices (ou voisinage).

Chaque itération de l'algorithme correspond à un déplacement de sa population de particules dans l'espace de recherche, selon trois typologies de comportements sociaux :

— le comportement inertiel, par lequel la particule tend à suivre le déplacement induit par sa vitesse ;

— le comportement cognitif, par lequel la particule va avoir tendance à se rapprocher de sa meilleure position visitée ;

— le comportement social, par lequel la particule se base sur les informations de son voisinage pour orienter son déplacement.

Le déplacement effectif de chaque particule, lors d'une itération de l'algorithme, est une combinaison pondérée et probabiliste de ces trois comportements. Il est illustré par la figure 1.8.

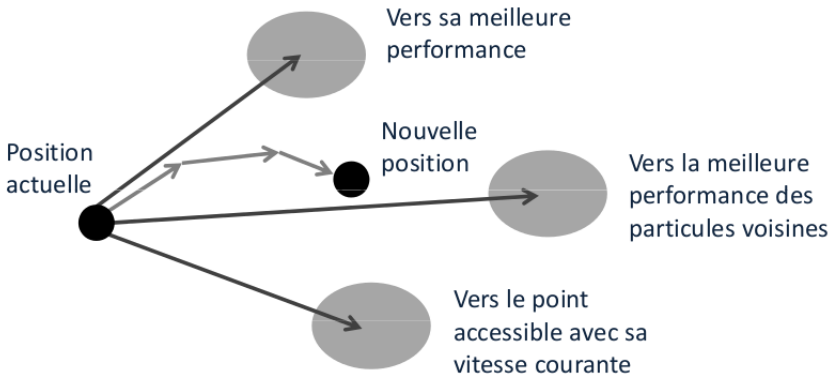

FIGURE 1.8 : Déplacement d'une particule [Lepagnot, 2011].

Considérant un problème de dimension D, N particules sont initialisées aléatoirement dans l'espace de recherche. La i^{ieme} particule est située à la position $X_i = (x_{i1}, \ldots, x_{iD})$, une solution du problème à optimiser. La qualité de la position de la particule est évaluée par la valeur de la fonction objectif en cette position. La i^{ieme} particule mémorise sa meilleure position atteinte jusqu'alors (en premier lieu, sa position initiale), notée $pbest_i$. Elle connaît également la meilleure position de son voisinage

notée $gbest_i$. Enfin, elle possède une vitesse initiale $V_i = (v_{i1}, \ldots, v_{iD})$.

Le déplacement de l'essaim est effectué à chaque itération de l'algorithme. Étant donné un état de l'essaim à un temps t, au temps $t+1$ la vitesse et la position de la i^{ieme} particule sont recalculées selon les équations (1.32) et (1.33) :

$$v_{i,j}^{t+1} = w\, v_{i,j}^{t} + c_1 r_{1_{i,j}}^{t} [pbest_{i,j}^{t} - x_{i,j}^{t}] + c_2 r_{2_{i,j}}^{t} [gbest_{i,j}^{t} - x_{i,j}^{t}],\ j \in \{1, \ldots, D\}. \tag{1.32}$$

$$x_{i,j}^{t+1} = x_{i,j}^{t} + v_{i,j}^{t+1},\ j \in \{1, \ldots, D\}. \tag{1.33}$$

où w est une constante, le *coefficient d'inertie* introduit en [Shi et Eberhart, 1998a,b], c_1 et c_2, deux constantes, les *coefficients d'accélération*. c_1 correspond à la composante *cognitive* de la particule car elle pondère le déplacement vers la meilleure position connue de la particule, c_2 correspond à la composante *sociale* car elle pondère le déplacement vers la meilleure position connue du voisinage de la particule. Enfin, r_1 et r_2 sont deux nombres aléatoires, tirés uniformément dans $[0, 1]$, pour chaque dimension.

L'algorithme 1.6 décrit le déroulement global de l'algorithme.

Algorithme 1.6 : Algorithme PSO

```
   // Phase d'initialisation de l'essaim
1 : Initialisation des N particules (vitesse v_i^0, position x_i^0 et pbest_i = x_i^0)
2 : Évaluation de la qualité des positions de chaque particule
   // Phase de déplacement de l'essaim
3 : tant que critère de fin non satisfait faire
4 :     red pour chaque particule i faire
5 :         mise à jour de pbest_i et gbest_i
6 :     fin
7 :     pour chaque particule i faire
8 :         déplacement de la particule selon (1.32) et (1.33)
9 :         évaluation de la qualité de la position
10 :    fin
11 :    Sauvegarde de la meilleure position de l'essaim
12 : fin
```

Les paramètres de l'algorithme sont nombreux : la taille de l'essaim, les différents coefficients w, c_1 et c_2, la topologie de voisinage utilisée. Différentes stratégies d'amélioration ont été mises en place pour modifier son comportement.

1.3.2.2 Améliorations

Au delà de sa simplicité formelle, les nombreuses caractéristiques de cet algorithme en font un algorithme puissant et fortement configurable. L'étude des différents coefficients comportementaux

permettant d'assurer la convergence, l'équation de base de la vitesse, les multiples possibilités de topologies de voisinage, l'évolution de la population sont autant de points d'intérêt des différents travaux de la communauté scientifique.

Vitesse et coefficients Lors d'un déplacement, une vitesse calculée trop élevée peut conduire une particule à sortir de l'espace de recherche. Dans [Eberhart et collab., 1996], Eberhart *et al.* ont introduit un paramètre contrôlant la vitesse V_{max} qui limite le déplacement à une distance maximum et empêche l'algorithme de diverger.

De plus, afin de confiner une composante de particule dans l'espace de recherche, plusieurs stratégies peuvent être introduites :

— La particule divergente est laissée à l'extérieur de l'espace de recherche mais son évaluation n'est pas réalisée. Elle ne pourra alors pas attirer d'autres particules en dehors de l'espace de recherche. Cette méthode est utilisée en optimisation sous contraintes ;

— la particule est stoppée à la frontière et la vitesse est annulée :
par exemple, si $x_{i,j}^{t+1} > x_{j_{max}}$, alors $x_{i,j}^{t+1} = x_{j_{max}}$ et $v_{i,j}^{t+1} = 0$;

— la particule « rebondit » sur la frontière, vers l'intérieur de l'espace de recherche :
par exemple, si $x_{i,j}^{t+1} > x_{j_{max}}$, alors $x_{i,j}^{t+1} = x_{i,j}^{t} + \alpha\ v_{i,j}^{t}$, $\alpha \in [-1, 0]$.

D'autres stratégies sont proposées par Clerc dans [Clerc, 2006a].

Plusieurs études ont été réalisées dans le but d'assurer la convergence de l'essaim et de résoudre la dichotomie entre exploration et exploitation [Van den Bergh, 2002; Kennedy et collab., 2001; Trelea, 2003]. Elles analysent les trajectoires des particules, l'évolution du coefficient d'inertie [Chatterjee et Siarry, 2006; Shi et Eberhart, 1999, 2001] ou encore la combinaison des paramètres w, c_1 et c_2.

Clerc et Kennedy, dans [Clerc et Kennedy, 2002], ont démontré que rendre ces paramètres dépendants conduit à une bonne convergence de l'algorithme. Ils introduisent un facteur de contrition χ qui permet de se passer du paramètre V_{max} pour contrôler la convergence de l'essaim. Cette variante de l'algorithme modifie l'équation (1.32) en :

$$v_{i,j}^{t+1} = \chi \left(v_{i,j}^{t} + \phi_1 r_{1_{i,j}}^{t} [pbest_{i,j}^{t} - x_{i,j}^{t}] + \phi_2 r_{2_{i,j}}^{t} [gbest_{i,j}^{t} - x_{i,j}^{t}] \right). \tag{1.34}$$

avec :

$$\chi = \frac{2}{|\phi - 2 + \sqrt{\phi^2 - 4\phi}|}, \text{ où } \phi = \phi_1 + \phi_2,\ \phi > 4. \tag{1.35}$$

Ils proposent comme valeurs de coefficients : $\phi = 4,1,\ \phi_1 = \phi_2$ et $\chi = 0,729844$. Le paramètre ϕ permet d'éviter une convergence prématurée, tout en assurant que l'algorithme converge vers un état d'équilibre. Il est intéressant de noter que l'équation (1.34) correspond à l'équation (1.32) avec

comme correspondance de coefficients $w = \chi = 0,729844$ et $c_i = \chi\ \phi_i = 1,49618$.

Bien que l'introduction du coefficient de contrition permette de se passer de l'utilisation du paramètre V_{max}, Eberhart et Shi dans [Eberhart et Shi, 2000] concluent que l'utilisation d'un paramètre $V_{i_{max}} = X_{i_{max}}$ conduit à de meilleurs résultats.

D'autres études sur l'assurance de la convergence de l'essaim sont exposées dans les articles [Van den Bergh, 2002; Van den Bergh et Engelbrecht, 2006; Trelea, 2003; Zheng et collab., 2003].

FIPS Une autre variante populaire est l'algorithme FIPS (pour *Fully Informed Particle Swarm*) [Mendes et collab., 2004]. FIPS modifie le comportement de l'algorithme pour l'équation de mise à jour de la vitesse. Dans la version de base de PSO, la meilleure particule ($pbest_i^t$) d'une topologie de voisinage définie intervient dans le calcul de la vitesse d'une particule i (éq. (1.32)). Mendes *et al.* ont proposé de ne pas se servir uniquement des informations du meilleur voisin et d'utiliser l'ensemble des particules composant le voisinage.

L'équation d'ajustement de la vitesse de la j^{ieme} composante de la particule i devient :

$$v_{i,j}^{t+1} = \chi \left(v_{i,j}^t + \sum_{n=1}^{N_i} \frac{U(0,\phi)(pbest_{n,j}^t - x_{i,j}^t)}{N_i} \right). \tag{1.36}$$

avec χ provenant de (1.33), $\phi = 4,1$ et N_i, la taille du voisinage de la particule i. Dans [Mendes et Neves, 2004], Mendes et Neves discutent de la structure de la population et de son influence dans FIPS. Une étude de convergence de l'algorithme FIPS peut être trouvée dans [Montes de Oca et Stützle, 2008].

***Bare bones* PSO** En 2003 [Kennedy, 2003], Kennedy propose de se passer de l'équation de la vélocité et propose une nouvelle version de l'équation de déplacement permettant de ne pas utiliser des composantes cognitives et sociales. Le déplacement est alors effectué selon une loi de probabilité suivant une distribution gaussienne de moyenne donnée par la moyenne entre les valeurs de la meilleure position personnelle *pbest* et de celle du voisinage *gbest*, et d'écart-type la valeur absolue de la différence entre ces deux positions (1.37) :

$$x_{i,j}^{t+1} = \mathcal{N}(\frac{pbest_j + gbest_j}{2},\ |gbest_j - pbest_j|),\ j \in \{1, \ldots, D\}. \tag{1.37}$$

Basées sur cet algorithme, des améliorations ont été proposées : [Blackwell, 2012; Campos et collab., 2014; Krohling et Mendel, 2009].

***Heterogeneous* PSO** Engelbrecht propose d'utiliser différents comportements de déplacement selon un ensemble défini [Engelbrecht, 2010]. Cet ensemble peut être soit statique, dans le sens où un

comportement est affecté à une particule lors de l'initialisation de l'essaim, puis n'est jamais modifié ; soit dynamique : la particule change de comportement lorsqu'elle n'améliore plus sa meilleure position.

Voisinage Un autre paramétrage important de l'algorithme concerne la topologie de voisinage d'une particule. En effet, dans l'équation (1.32), la valeur *gbest* est la meilleure valeur donnée par l'ensemble des particules informatrices appartenant au voisinage de la particule considérée.

Dans la première version de l'algorithme, la topologie de voisinage appliquée est globale (dite *Gbest*). Une particule est informée par l'ensemble des particules formant l'essaim. Cela revient à partager l'information de l'optimum global de l'essaim trouvé jusqu'alors. La topologie *Gbest* a l'avantage de converger rapidement, mais présente l'inconvénient de pas explorer suffisamment l'espace de recherche, ce qui a pour conséquence de converger vers un optimum local [Shi et Eberhart, 1999]. Un autre type de topologie est la topologie locale (ou *Lbest*), où une particule est informée par un sous-ensemble de ses congénères. De nombreux types de topologies locales, statiques ou dynamiques, ont été proposés dans la littérature. Une version « unifiée », présentée dans [Parsopoulos et Vrahatis, 2007, 2005], définit un compromis entre les topologies *Lbest* et *Gbest*, en appliquant une pondération aux vitesses obtenues dans chaque contexte pour arriver à une nouvelle topologie.

Le voisinage défini est généralement un voisinage social, sans contrainte de localisation des voisins, même si certains travaux vont dans ce sens comme dans [Lane et collab., 2008] où une triangulation de Delaunay est effectuée pour construire le voisinage géographique.

Eberhart *et al.* ont proposé une première version suivant une topologie statique en anneau [Eberhart et Kennedy, 1995]. Dans [Kennedy, 1999], Kennedy a montré que des voisinages restreints, aléatoires peuvent donner des résultats intéressants sur les performances de l'algorithme. De nombreuses études ont été menées, définissant chacune une typologie de topologie.

Dans [Kennedy et Mendes, 2002; Mendes et collab., 2004], les auteurs présentent notamment une topologie en *étoile*, où l'information passe par une unique particule reliée à toutes les autres, la topologie de *Von Neumann* dans laquelle les particules sont organisées sous forme de grille, chacune possédant quatre voisins, *four-clusters*, où quatre voisinages communiquent deux à deux par un couple de particules informatrices, ou bien encore une topologie *pyramide*. La figure 1.9 illustre ces topologies.

Bien que ces topologies améliorent les résultats de l'algorithme PSO, elles ne résolvent pas le problème de convergence prématurée.

Des topologies dynamiques ont alors été proposées afin de permettre à l'essaim de modifier sa structure. Cela influence son déplacement et permet son extraction hors d'un optimum local. Parmi les topologies dynamiques proposées, [Clerc, 2006b] suggère une topologie aléatoire, dans laquelle une particule est informée par un nombre aléatoire de voisines, entre 1 et S. Pour chaque particule, le voisinage est redéfini si la solution globale n'est pas améliorée. Dans [Lim et Isa, 2014], une connectivité

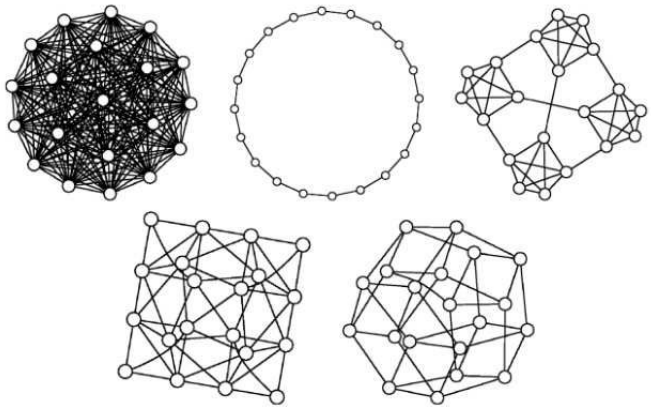

FIGURE 1.9 : Exemples de topologies : *Gbest*, *en anneau*, *four-clusters*, *Von Neumann* et *pyramide* [Kaveh, 2014].

croissante entre particules est présentée. Chaque particule est reliée à une autre particule, de manière unidirectionnelle. Au cours du déroulement de l'algorithme, à intervalle de temps défini, des arêtes sont ajoutées pour augmenter la connectivité du graphe. Un mélange aléatoire des arêtes est aussi effectué. À l'inverse, dans [Montes de Oca et collab., 2009], les auteurs proposent l'utilisation de l'algorithme FIPS avec une topologie de voisinage évolutif dans lequel toutes les particules sont interconnectées, au temps $t = 0$. Un cycle en anneau étant défini, au cours du temps, des arêtes n'appartenant pas à ce cycle sont supprimées jusqu'à converger vers la topologie en anneau définie. De même, Mohais *et al.* [Mohais et collab., 2005] proposent une approche utilisant FIPS, mais pour celui-ci, les voisinages initialement aléatoires se restructurent au cours du temps en modifiant seulement les arêtes et en gardant une taille fixe. [Wang et Xiang, 2008] proposent une structure de voisinage de type anneau, unidirectionnelle, à l'intérieur de laquelle les particules sont classées selon une valeur de *fitness*, la plus grande valeur de *fitness* correspond à la première particule, la plus « mauvaise ». Le tri de la structure est rafraîchi à chaque mise à jour. Dans [El Dor et Siarry, 2015], les auteurs présentent une topologie, *Dcluster*, qui combine la topologie statique *four clusters* et la topologie dynamique *fitness*. Les clusters sont organisés selon la *fitness* des particules. Un cluster central contient les plus « mauvaises » particules et chacune de ces particules communique avec la plus « mauvaise » particule de chacun des autres clusters. Une étude plus détaillée de différents types de voisinage est consultable dans la thèse d'Abbas El Dor [El Dor, 2012].

Tribes Une des contraintes de l'algorithme PSO est le nombre important de paramètres de l'algorithme : le nombre de particules, les différents coefficients, le type de voisinage, etc. En effet, même si certaines valeurs sont conseillées, il est illusoire de penser qu'elles sont génériques. M. Clerc a proposé un algorithme d'optimisation par essaim particulaire sous forme de boîte noire, l'algorithme TRIBES [Clerc, 2003]. L'objectif est de fournir un algorithme robuste, sans paramètre à régler et efficace dans

la plupart des cas.

Dans cet algorithme, l'essaim est divisé en sous-essaims, les « tribus », de tailles différentes, évoluant durant l'exécution de l'algorithme. Chaque sous-essaim explore localement une région, puis une décision globale est prise.

Les sous-essaims effectuent deux types de communications : interne et externe. Les communications internes correspondent à l'algorithme classique pour un sous-essaim. Les communications externes sont effectuées entre deux sous-essaims. L'algorithme introduit une composante qualitative d'une particule en évaluant les deux derniers déplacements. De plus, le nombre de particules composant un essaim évolue au cours du temps. Un sous-essaim composé de bonnes particules aura tendance à supprimer la pire, alors que des sous-essaims composés de particules de moins bonne qualité, ou évoluant peu, créeront chacun une particule, formant alors un nouvel essaim. Le nombre de particules initial est alors de 1. Pour supprimer les paramètres correspondant aux coefficients d'inertie et d'accélération, une nouvelle équation de la vitesse est proposée :

$$v_{i,j}^{t+1} = \chi \left(v_{i,j}^{t} + U(0,\phi/2)(pbest_{i,j}^{t} - x_{i,j}^{t}) + U(0,\phi/2)(gbest_{i,j}^{t} - x_{i,j}^{t}) \right). \tag{1.38}$$

La thèse de Yann Cooren [Cooren, 2008] propose une étude poussée de cet algorithme ainsi que plusieurs améliorations.

D'autres approches d'optimisation coopérative, locale/globale, à plusieurs essaims sont abordées dans la littérature. Dans [Liang et Suganthan, 2005; Xu et collab., 2015], différents sous-essaims naviguent dans tout l'espace de recherche puis des regroupements de particules sont effectués pour reformer de nouveaux sous-essaims. Niu *et al.* [Niu et collab., 2007] utilisent un principe d'essaims maîtres/auxiliaires. Les sous-essaims auxiliaires résolvent l'algorithme durant plusieurs itérations et partagent leur meilleur individu avec l'essaim maître, qui fait ensuite évoluer ses particules. Ce processus est alors itéré. Dans [Van den Bergh et Engelbrecht, 2004], les auteurs utilisent plusieurs essaims pour optimiser une sous-partie de l'espace de recherche. L'espace de recherche de dimension D est divisé en sous-espaces de plus petite dimension, chaque essaim s'occupant d'optimiser une sous-partie du problème. Une autre approche de partition de l'espace de recherche est proposée dans [El Dor et Siarry, 2015]. L'espace de recherche est divisé en zones et chaque sous-essaim n'explore qu'une partie locale, puis un nouvel essaim est construit avec l'ensemble des meilleures solutions.

1.3.3 Algorithme de Colonie d'Abeilles Artificielles

L'algorithme de colonie d'abeilles artificielles ou *Artificial Bee Colony* (ABC) a été introduit par Dervis Karaboga [Karaboga, 2005] et développé depuis 2005 par Karaboga et Basturk [Karaboga et Basturk, 2008] pour les problèmes d'optimisation continue. C'est un algorithme à population,

d'inspiration naturaliste, basé sur le butinage des abeilles.

1.3.3.1 Principe

Dans cet algorithme, une solution candidate au problème d'optimisation est représentée par une source de nourriture. Chaque source de nourriture possède une quantité de nectar qui caractérise sa qualité (*fitness*).

La population de la colonie est divisée en trois groupes d'abeilles, parcourant l'espace de recherche en quête de source de nourriture. Contrairement à l'algorithme précédent, ces groupes d'abeilles ne représentent pas les solutions mais un ensemble d'itérations. Chaque groupe d'itérations correspond au vol d'un des trois types d'abeilles : les abeilles *ouvrières*, les *spectatrices* et les *exploratrices*.

Les ouvrières parcourent le voisinage des sources de nourriture afin de trouver une meilleure source que celle visitée. Elles partagent ensuite la qualité de la source avec les spectatrices. Ces dernières se concentrent essentiellement sur les sources de nourriture de meilleure qualité. Lorsqu'une source de nourriture a été suffisamment explorée, elle est abandonnée et les exploratrices partent aléatoirement à la recherche d'une nouvelle source.

L'algorithme 1.7 résume le processus.

Le nombre d'abeilles ouvrières et de spectatrices correspond au nombre de sources de nourriture. Il y a généralement une abeille exploratrice. Une source de nourriture est un vecteur de dimension D, D étant la dimension du problème, le nombre de sources de nourriture (SN) est un paramètre de l'algorithme.

Initialisation Premièrement, l'algorithme initialise une population de SN individus, comme décrit en équation (1.39), X_{min} et X_{max} sont les vecteurs des valeurs minimum et maximum de chaque dimension du problème.

$$\begin{aligned} & i \in \{1, \ldots, SN\},\ j \in \{1, \ldots, D\} \\ & X_j^i \ = \ X_{min_j} + rand[0,1](X_{max_j} - X_{min_j}). \end{aligned} \tag{1.39}$$

À chaque source de nourriture est associée une quantité de nectar définissant une attractivité, la *fitness*. Cette valeur est calculée selon l'équation (1.40), où f est la fonction objectif.

$$fit(X^i) \ = \ \begin{cases} \frac{1}{f(X^i)+1} & , \ f(X^i) \ \geq \ 0, \\ \\ 1 + |f(X^i)| & , \ f(X^i) \ < \ 0. \end{cases} \tag{1.40}$$

Algorithme 1.7 : Algorithme ABC

Initialisation des sources de nourriture (1.39) et de leur *fitness* (1.40)
tant que *critère de fin non satisfait* **faire**
    `// Vol des ouvrières`
    pour $i = 1$ *à* $nbOuvrières$ **faire**
        génération d'une nouvelle solution (source de nourriture) N^i selon (1.41)
        si $fit(N^i) > fit(X^i)$ **alors**
            $X^i \leftarrow N^i$
        fin
        sinon
            $nbVisites_i \leftarrow nbVisites_i + 1$
        fin
    fin
    Mise à jour de la qualité des sources de nourriture (*fitness*), selon (1.40).
    `// Vol des spectatrices`
    $i \leftarrow 1$
    tant que $i \leq nbSpectatrices$ **faire**
        $k \leftarrow rand(SN)$
        si $rand(0,1) < p_k$ *(cf (1.43))* **alors**
            génération d'une nouvelle solution N^k selon (1.41),
            si $fit(N^k) > fit(X^k)$ **alors**
                $X^k \leftarrow N^k$
            fin
            sinon
                $nbVisites_k \leftarrow nbVisites_k + 1$
            fin
            $i \leftarrow i + 1$
        fin
    fin
    `// Vol des exploratrices`
    pour $i = 1$ *à* $nbExploratrices$ **faire**
        Si la source la plus visitée est abandonnée, remplacement par une nouvelle solution, selon (1.39).
    fin
    Sauvegarde de la meilleure source de nourriture.
fin

Vol des ouvrières Chaque abeille ouvrière produit une nouvelle solution, au voisinage d'une source de nourriture existante, selon (1.41). Pour la i^{ieme} abeille ($i \in \{1, ..., SN\}$), $j \in \{1, 2, \ldots, D\}$, une dimension choisie aléatoirement, ϕ_{ij} est un réel aléatoire, uniformément distribué dans l'intervalle $[-1; 1]$ et $k \in \{1, \ldots, SN\}$, une solution choisie aléatoirement ($k \neq i$).

$$N_j^i = X_j^i + \phi_{ij}(X_j^i - X_j^k). \tag{1.41}$$

Si la nouvelle source de nourriture N^i ainsi produite possède une meilleure *fitness* que X^i, elle la remplace. Sinon, un compteur de nombre de visites infructueuses est incrémenté.

Vol des spectatrices Cette série d'itérations est semblable à la précédente, sauf qu'une abeille spectatrice visitera une source de nourriture prometteuse, qui possède une bonne quantité de nectar. Cette information est partagée par les ouvrières en utilisant la valeur de *fitness* (1.40). Elle permet de calculer une attractivité potentielle sous forme de probabilité de sélection. Il existe plusieurs façons de calculer cette probabilité p_i, comme décrit par les équations (1.42) et (1.43), où $fit(X^i)$ est la *fitness* de la i^{ieme} solution et fit_{max} la valeur de *fitness* maximale pour toute la population.

$$p_i = \frac{0,9\ fit(X^i)}{fit_{max}} + 0,1. \tag{1.42}$$

$$p_i = \frac{fit(X^i)}{\sum_{k=1}^{SN} fit(X^k)}. \tag{1.43}$$

Lors d'une itération d'abeille spectatrice, une nouvelle solution N^i est générée de la même manière que lors d'une itération d'abeille ouvrière, en utilisant l'équation (1.41).

Vol des exploratrices Lors de ces deux précédentes phases, lorsqu'une nouvelle solution N^i, générée à partir d'une solution X^i, n'améliore pas cette dernière, un compteur de visites $nbVisites_i$ est incrémenté pour la solution X^i. Lorsque ce compteur atteint une valeur *limite*, la solution est abandonnée et remplacée par une nouvelle solution générée selon (1.39).

L'avantage de cet algorithme est qu'il ne possède que peu de paramètres : le nombre de sources de nourriture, le nombre d'abeilles exploratrices et le nombre maximal de visites non améliorantes sur une source de nourriture.

1.3.3.2 Améliorations

Le déplacement unidimensionnel de l'algorithme ABC induit un faible taux de convergence ; de plus, comme les autres métaheuristiques, il souffre de convergence prématurée. Par ailleurs les problématiques étudiées se consacrent principalement à la résolution de ces deux questions, mais aussi à l'optimisation de ses différentes caractéristiques (paramètres, différentes phases, équation de base, probabilité de sélection...), ou encore, comme les autres métaheuristiques, à la dichotomie entre exploration et exploitation. Plusieurs études de l'algorithme proposent des solutions pour améliorer ces différents points.

Paramètres Dans [Karaboga et Basturk, 2008], les auteurs suggèrent de sélectionner le nombre SN de sources de nourriture dans l'intervalle $[50, 100]$; le compteur de visites infructueuses dépend de la dimension D du problème et de la taille de la population N selon $limit = ND$ ou bien $limit = ND/2$; enfin la composante stochastique de l'équation (1.41), ϕ_{ij}, est tirée aléatoirement entre $[-1, 1]$. Diwold *et al.* suggèrent que la taille de la population est fortement liée au problème d'optimisation. Néanmoins, des valeurs élevées pour la taille de la population fournissent de meilleurs résultats. La même observation de dépendance concernant le facteur *limit* est faite et les auteurs laissent la question ouverte.

Dans [Qin et collab., 2015], les auteurs suggèrent de modifier au cours du temps, linéairement ou non, le ratio entre la population d'ouvrières et de spectatrices. Dans le cas non linéaire, donnant les meilleurs résultats, le ratio du nombre d'ouvrières est donné en fonction du nombre d'évaluations par : $r_c = r_{max} - (r_{max} - r_{min}) \left(\frac{nb_{eval}}{max_{eval}}\right)^{\alpha}$, où l'on adopte les valeurs empiriques suivantes : $r_{max} = 0,7$, $r_{min} = 0,2$ et $\alpha = 1,2$; nb_{eval} est le nombre d'évaluations en cours, max_{eval}, le nombre d'évaluations maximum défini.

Équation de déplacement Afin d'améliorer l'exploitation de l'algorithme, Zhu et Kwong [Zhu et Kwong, 2010] s'inspirent de PSO en incluant le meilleur individu dans l'équation de déplacement (1.41). Une pondération de l'information du meilleur individu trouvé jusqu'alors est intégrée dans l'équation de base. Dans leur algorithme, GABC, l'équation (1.44) est utilisée pour les phases ouvrières et spectatrices.

$$V_j^i = X_j^i \; + \; \phi_{ij}(X_j^i - X_j^k) \; + \; \psi_{ij}(X_j^{best} - X_j^i), \; k \neq i. \tag{1.44}$$

ψ_{ij} est un nombre aléatoire dans $[0, C]$, la valeur conseillée pour C est : $1,5$.

Li *et al.* proposent I-ABC [Li et collab., 2012], un autre algorithme inspiré de PSO et du précédent GABC. En effet, l'équation de déplacement (1.45) fait intervenir des coefficients d'inertie w_{ij} et d'ac-

célération Φ_1, Φ_2 contrôlant l'amplitude du déplacement.

$$V_j^i = w_{ij}\ X_j^i\ +\ 2(\phi_{ij} - 0,5)(X_j^i - X_j^k)\Phi_1\ +\ \psi_{ij}(X_j^{best} - X_j^i)\Phi_2,\ k \neq i. \tag{1.45}$$

ϕ_{ij} et ψ_{ij} sont deux nombres aléatoires entre $[0, 1]$. Les valeurs des coefficients d'accélération sont caractérisées par la $fitness$ de la solution $(fit(X^i))$ et celle de la meilleure solution à l'initialisation (ap).

$$w_{ij} = \Phi_1 = \frac{1}{1 + exp(-fit(X^i)/ap)^{iter}}. \tag{1.46}$$

$$\Phi_2\ =\ \begin{cases} 1 & ,\ \text{vol des ouvrières} \\ \frac{1}{1+exp(-fit(X^i)/ap)^{iter}} & ,\ \text{vol des spectatrices.} \end{cases} \tag{1.47}$$

Dans le même article, suite au constat d'une faiblesse de l'algorithme sur certaines fonction, face à ABC et GABC, une version PS-ABC est présentée, dans laquelle les abeilles fournissent trois solutions basées sur les équations (1.41), (1.44) et (1.45), pour ensuite sélectionner celle de la meilleure *fitness*.

Ce mimétisme avec PSO est aussi observé dans [Imanian et collab., 2014], où les auteurs modifient l'équation des spectatrices par celles de PSO (1.32) et (1.33). Pour cela, ils incluent dans la solution une mémoire de la meilleure position personnelle.

Le principe d'inclure la meilleure solution est proposé dans [Banharnsakun et collab., 2011]. Banharnsakun *et al.* modifient le mouvement des abeilles spectatrices, basé sur le mouvement de la meilleure ouvrière précédemment calculé :

$$V_j^i = X_j^i\ +\ \phi_{ij}\ fit(X^{best})\ (X_j^i - X_j^{best}). \tag{1.48}$$

ϕ_{ij} est un nombre tiré aléatoirement entre $[-1, 1]$, $fit(X^{best})$ est la valeur de *fitness* de la meilleure solution X^{best}. Ils proposent aussi une nouvelle équation de génération de solution (1.49) pour la phase des exploratrices :

$$V_j^i = X_j^i\ +\ \phi_{ij}\left[w_{max} - \frac{nb_{eval}}{max_{eval}}(w_{max} - w_{min})\right] X_j^i,\ j = 1 \ldots D. \tag{1.49}$$

où w_{max} et w_{min} sont des pourcentages d'ajustement, fixés respectivement à 1 et $0,2$. À noter que, pour le remplacement de la solution courante, les auteurs modifient le critère de sélection. Ils ne se basent plus sur la $fitness$ mais sur l'évaluation par la fonction objectif de la solution (i.e. pour une fonction f : Si $f(V^i) < f(X^i)$ alors $X^i \leftarrow V^i$).

D'autres papiers incluent la meilleure solution dans leur équation de déplacement. Dans [Sulaiman et collab., 2013], une nouvelle phase est incluse avant celle des exploratrices, dans laquelle un nombre défini de solutions non satisfaisantes est redirigé vers la meilleure solution globale. De nombreux tra-

vaux s'intéressent aux schémas de mutation de l'algorithme d'évolution différentielle afin d'améliorer la vitesse de convergence (déplacements multi-directionnels) ainsi que l'exploitation (inclusion de la meilleure solution, comme vu précédemment). Fister *et al.* [Fister et collab., 2012] utilisent différentes équations de recherche pour chacune des phases de l'algorithme. Pour les ouvrières et les spectatrices, les équations sont inspirées des schémas de mutation de DE : le schéma *DE/rand/1/bin* (éq. (1.1)) est utilisé lors du vol des ouvrières, *DE/current-to-best/1/bin* (éq. (1.10)), faisant intervenir la meilleure solution. Les auteurs modifient de plus le calcul de probabilité en faisant intervenir les évaluations des meilleures et plus mauvaises solutions. Kiran *et al.* [Kiran et collab., 2015] utilisent aussi un pool de déplacement inspiré de l'évolution différentielle. Lorsqu'un schéma est sélectionné, un compteur de réussite est incrémenté. Le ratio de ce compteur sur la somme de tous les compteur donne une probabilité évolutive de sélection. Les auteurs de [Wang et collab., 2013], à la manière de l'algorithme JADE, utilisent une archive des m meilleures solutions stockées.

Dans [Akay et Karaboga, 2012], Akay et Karaboga proposent d'améliorer la vitesse de convergence, due au déplacement unidimensionnel, en incluant un paramètre, le taux de modification. Ce taux correspond à la probabilité qu'une dimension soit modifiée. Ce paramètre MR permet de modifier plus d'une dimension à la fois. L'équation de génération de l'algorithme ABC originel (1.41) devient alors :

$$V_j^i = \begin{cases} X_j^i + \phi_{ij}(X_j^i - X_j^k) & , \text{ si } r_j < \text{MR}, \\ X_j^i & , \text{ sinon.} \end{cases} \tag{1.50}$$

Pour chaque dimension j, deux nombres aléatoires sont fournis : $r_j,\ 0 \leq r_j \leq 1$ et $\phi_{ij} \in [-1;1]$. Si, pour la dimension en cours, le nombre aléatoire r_j est plus petit que le taux de modification, le décalage est appliqué à cette dimension. À chaque itération du processus de recherche, un sous-ensemble de dimensions de la solution X^i est alors modifié pour engendrer le nouvel individu. De plus, les auteurs introduisent un pas adaptatif de déplacement SF (*Scaling Factor*). Cette valeur permet de définir un nouvel intervalle : $[-SF, SF]$, pour le tirage aléatoire réalisé dans l'équation (1.41). Ce pas est mis à jour de manière cyclique selon (1.51).

$$SF_{t+1} = \begin{cases} 0,85\ SF_t & , \ Si\ \phi(m) < 1/5, \\ SF_t\ /\ 0,85 & , \ Si\ \phi(m) > 1/5, \\ SF_t & , \ Si\ \phi(m) = 1/5. \end{cases} \tag{1.51}$$

où $\phi(m)$ est le ratio de déplacements améliorants, m est le nombre d'évaluations. Les auteurs montrent que les valeurs de ces coefficients dépendent du type de problème. Un taux de modification $MR = 0,4$ fournit des résultats performants. Le pas de déplacement donnant globalement les meilleurs résultats est $SF = 1$, correspondant à la version d'origine.

Basée sur le principe du taux de modification et dans le but d'accélérer une nouvelle fois la convergence, une stratégie adoptée dans [Kiran et Findik, 2015] est de comparer la précédente valeur de *fitness* de la solution et la nouvelle, afin de mettre à jour une direction de déplacement par dimension, d_{ij}. Si la nouvelle solution est de moins bonne qualité (selon sa valeur de *fitness*), l'information de direction est mise à 0 pour toutes les dimensions. Le cas échéant, chaque direction de dimension est mise à jour à -1 si la valeur de la dimension de l'ancienne solution est plus petite que la nouvelle, à 1 sinon. L'équation de déplacement est alors remplacée par (1.52)

$$V_j^i = \begin{cases} X_j^i + \phi_{ij}\ (X_j^i - X_j^k) & , \text{ si } d_{ij} = 0, \\ X_j^i + d_{ij}\ rand(0,1)\ |X_j^i - X_j^k| & , \text{ sinon.} \end{cases} \tag{1.52}$$

où d_{ij} est la direction de déplacement, dans $\{-1,\ 0, 1\}$. La méthode, testée sur des fonction séparables, fournit de bons résultats.

Voisinage et recherches locales Dans [Jadon et collab., 2014], s'inspirant de l'algorithme DEGL de Das *et al.* [Das et collab., 2009], les auteurs proposent d'utiliser deux voisinages, l'un local et l'autre global, afin d'engendrer une nouvelle solution lors du vol des ouvrières. La méthode de création d'un nouvel individu est la même que celle présentée en section 1.2.2, équations (1.13), (1.14) et (1.15).

Karaboga et Gorkemli [Karaboga et Gorkemli, 2014] ont proposé une nouvelle méthode de déplacement par la recherche du meilleur individu dans le voisinage local de la solution à déplacer (1.53). Cette méthode est utilisée lors du vol des spectatrices, pour améliorer le caractère d'exploitation de l'algorithme.

$$V_j^i = X_j^{best} + \phi_{ij}(X_j^{best} - X_j^k),\ k \neq i. \tag{1.53}$$

Le voisinage est formé par l'ensemble des solutions proches de la solution en cours selon un paramètre r défini par : $Si\ d(X^i, X^m) < r\ d_{moy},\ alors\ X^m\ est\ un\ voisin\ de\ X^i$; d_{moy} est la distance euclidienne moyenne entre X^i et les autres solutions de la population. Le critère définissant la meilleure solution est basé sur la *fitness* et non sur l'évaluation.

Plusieurs auteurs s'appliquent à améliorer la phase exploitation de l'algorithme, en réalisant une recherche locale autour de la meilleure solution. La phase de recherche mémétique est généralement réalisée en fin de cycle (après le vol des exploratrices), sur la meilleure solution. Parmi ces travaux, nous pouvons citer Bansal *et al.* qui proposent *Memetic* ABC (MeABC) [Bansal et collab., 2013], un algorithme utilisant la méthode du nombre d'or [Kiefer, 1953]. Cette recherche utilise la méthode du nombre d'or jusqu'à un critère d'arrêt et vise à obtenir des valeurs de ϕ_{ij} permettant à l'équation (1.41) de trouver des solutions améliorantes autour de la meilleure solution globale. Fister *et al.* [Fister et collab., 2012] emploient aléatoirement deux techniques de recherches locales (les algorithmes du

simplexe de Nelder-Mead [Nelder et Mead, 1965] et de marche aléatoire avec exploitation de direction [Rao et Rao, 2009]) pour améliorer la meilleure solution. Dans [Chen et collab., 2012], les auteurs utilisent l'algorithme du recuit simulé [Kirkpatrick et collab., 1983] avec comme individu la meilleure solution.

Une état de l'art plus exhaustif peut être trouvé dans les travaux de Karaboga *et al.* [Karaboga et collab., 2012] ainsi que dans la thèse de Jadon [Jadon, 2015].

1.4 Généralités sur l'analyse de sensibilité

Nous allons par la suite recenser les principales méthodes d'analyse de sensibilité en vue de leur intégration dans les métaheuristiques.

Plusieurs métaheuristiques à population ont été détaillées dans la section précédente. Nous avons étudié leurs différentes caractéristiques ainsi qu'un certain nombre d'améliorations de chacune d'entre elles. Nous avons constaté qu'une des problématiques clés pour chacun de ces algorithmes est la gestion de l'équilibre entre exploration et exploitation. En d'autres termes, comment parcourir l'espace de recherche efficacement afin de découvrir une nouvelle zone prometteuse, ou en exploiter une prometteuse déjà découverte.

Dans le contexte de l'optimisation continue, un tel parcours de l'espace de recherche est intrinsèquement complexe. En effet, considérant une variable, entre deux valeurs de cette variable, il existe théoriquement une infinité de valeurs. Si l'on considère l'ensemble des dimensions du problème, cela complexifie considérablement cette tâche.

La malédiction de la dimension est un terme, introduit par Bellman [Bellman, 1957], qui désigne le problème d'accroissement exponentiel du volume de l'espace lorsque la dimension de celui-ci augmente.

La figure 1.10 [Keogh et Mueen, 2010] illustre ce principe. En dimension $D = 2$, si l'on considère le carré de côté 2, centré en un point et le cercle inscrit dans le carré, de rayon 1, le ratio de la surface du carré couverte par le cercle est $r = \frac{\pi\, 1^2}{2^2} = 78,5\%$. À mesure que le nombre de dimensions augmente, ce ratio diminue de manière drastique : pour $D = 3$, $r = 51,8\%$, pour $D = 6$, $r = 8\%$ et pour $D = 10$, $r = 0,2\%$.

Dans le cadre de l'optimisation, la recherche de voisinage et le déplacement d'une solution sont fortement impactés. Plus le nombre de dimensions augmente, plus l'espace entre les individus d'une population est important (cf Fig. 1.11).

Dans le contexte de l'exécution d'une métaheuristique, il serait pertinent de pouvoir se déplacer de manière efficace dans un espace de recherche, afin d'éviter des évaluations superflues, qui n'apporteront pas ou peu d'amélioration au résultat.

1. http ://cleverowl.uk/2016/02/06/curse-of-dimensionality-explained/

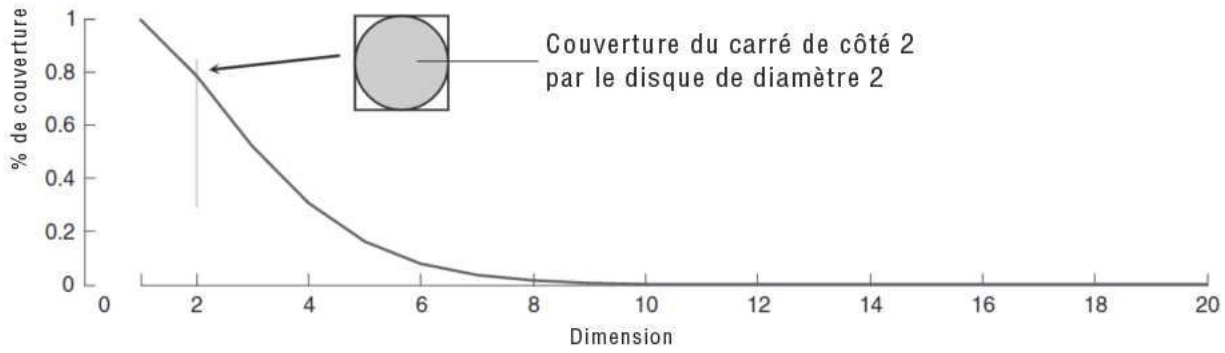

FIGURE 1.10 : Évolution du ratio de couverture de l'hypercube par l'hypersphère.

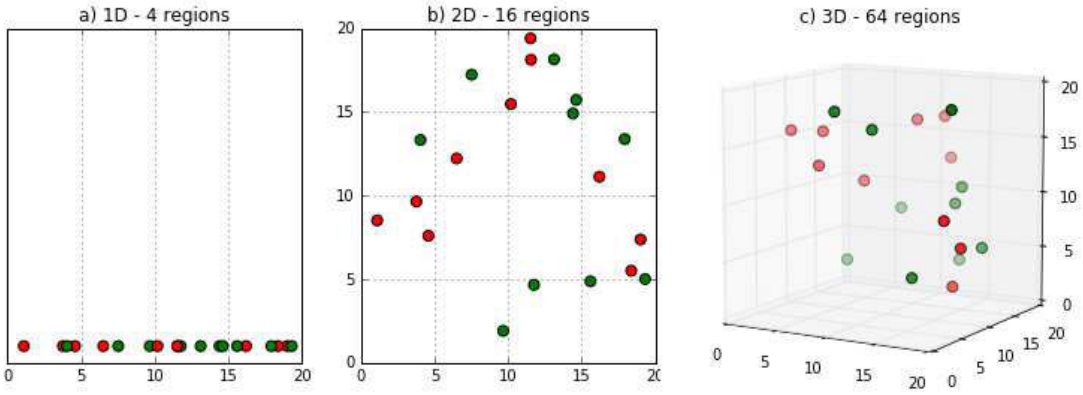

FIGURE 1.11 : La malédiction de la dimension (source : Clever OWL [1]).

Un problème d'optimisation est composé de variables d'entrée, $X_1, ..., X_D$, aléatoires, indépendantes et uniformément distribuées dans leur espace de variation. L'évaluation Y d'une fonction objectif f, $Y = f(X_1, ..., X_D)$, dépendante de ces variables d'entrée, est alors aussi considérée comme étant une variable aléatoire. Les méthodes d'analyse de sensibilité étudient l'impact de la variation des variables d'entrée $X_1, ..., X_D$ sur la variation Y.

L'analyse de sensibilité (SA) permet de déterminer quelles variables d'un modèle (i.e. code de calcul, fonction d'évaluation) possèdent une influence significative sur ce modèle. Saltelli, dans [Saltelli, 2002b], la définit comme suit : « L'analyse de sensibilité est l'étude de comment l'incertitude d'un modèle (numérique ou autre) peut être expliquée par les différentes sources d'incertitude en entrée du modèle ». Par incertitude, on entend la variabilité d'une donnée. L'idée sous-jacente à cette définition est l'étude de l'effet de la variation d'une variable d'entrée sur la réponse du modèle. L'analyse de sensibilité permet d'identifier quelle variable possède une forte influence sur la réponse et inversement, quelle variable a une influence moindre. Elle permet également d'explorer le modèle afin d'en déduire ses propriétés (linéarité, non linéarité, corrélation). Elle est habituellement utilisée en simulation

et/ou en modélisation, avec plusieurs domaines d'application, comme l'étude de la climatologie, les dynamiques de population, la biologie, la macroéconomie, la gestion du trafic, etc.

Cette étude se réalise en quatre étapes :

— le choix d'une distribution de probabilité uniforme, gaussienne ou autre, permettant de choisir les points à considérer dans l'étude ;
— l'échantillonnage, la sélection aléatoire, selon différentes méthodes, comme les plans d'expérience, le tirage aléatoire, de plusieurs points de l'espace ;
— l'évaluation des réponses du modèle pour l'échantillon réalisé ;
— le calcul des indices de sensibilité, en utilisant différentes approches, telles que le calcul des indices de Sobol ou des effets élémentaires que l'on abordera dans la suite du document.

L'analyse de sensibilité peut être locale ou globale. Une étude locale s'apparente au calcul des différentes dérivées partielles en un point. Elle permet d'identifier la direction de la plus forte variation autour d'une zone d'intérêt de l'espace de variation. Une étude globale s'intéresse à la variabilité de la réponse engendrée par la variation des différentes variables sur l'ensemble de leur domaine de définition.

Les *indices de sensibilité* définissent une mesure de cette variabilité. La valeur d'un indice est proportionnelle à son influence sur la sortie. Un exemple d'indice défini par Bauer et Hambi [Bauer et Hamby, 1991] est donné par :

$$I_{x_i} = \frac{max_{x_i}\left[f(x_1,\ldots,x_i,\ldots,x_D)\right] \; - \; min_{x_i}\left[f(x_1,\ldots,x_i,\ldots,x_D)\right]}{max_{x_i}\left[f(x_1,\ldots,x_i,\ldots,x_D)\right]}, \; \forall i \in \{1,\ldots,D\} \qquad (1.54)$$

Pour une entrée donnée, on calcule les réponses pour un ensemble de valeurs de x_i, gardant les autres fixes. L'indice de la i^{ieme} variable est calculé selon les valeurs minimum et maximum identifiées. Cet indice est néanmoins trop dépendant des valeurs choisies pour être précis. D'autres indices sont utilisés, tels que les *indices de sensibilité principal et total*, abordés en 1.8.

Dans [Iooss et Lemaître, 2015; Saltelli, 2002b], les auteurs dressent une liste de plusieurs approches d'analyse de sensibilité : locale (*One-At-Time* (OAT)), criblage (*screening*) par discrétisation de l'espace de recherche (dont la méthode de Morris), approximation par métamodèle ou encore globale (basée sur la variance). Ces méthodes sont abordées par la suite. La figure 1.12 [Iooss et Lemaître, 2015] cartographie les différentes méthodes d'analyse de sensibilité. Elle présente les méthodes conseillées, selon le type d'incertitude du modèle (fonction linéaire, non linéaire) et le nombre d'évaluations à effectuer pour réaliser l'analyse (proportionnellement au nombre de variables D du modèle).

Nous allons maintenant présenter les principales méthodes d'analyse de sensibilité.

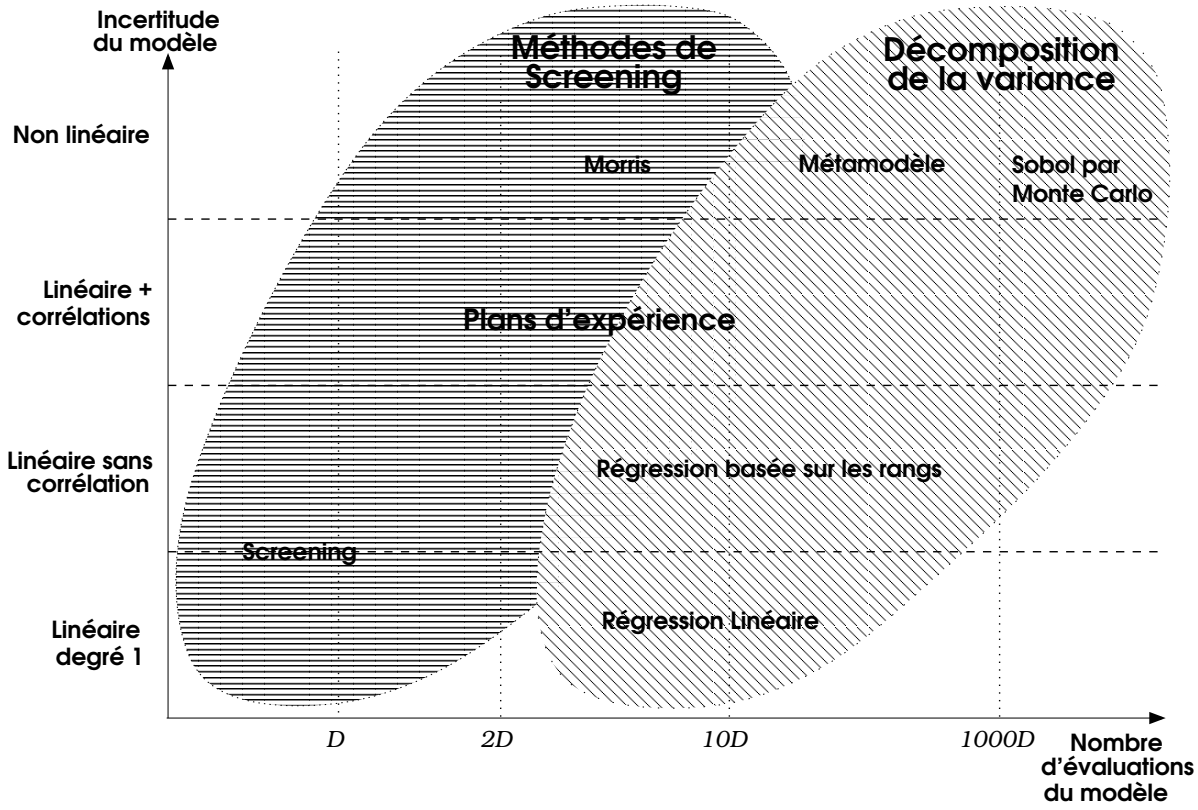

FIGURE 1.12 : Cartographie des méthodes d'analyse de sensibilité [Iooss et Lemaître, 2015].

1.5 Méthodes basées sur la régression linéaire

Pour un modèle de dimension D, on considère un ensemble de N points $X^i = (X_1^i, ..., X_D^i)$ et N évaluations d'un modèle Y^i, $i = (1, ..., N)$. Parmi les méthodes basées sur la régression, nous présentons ci-après deux méthodes faisant sens dans un contexte linéaire ou bien monotone. Les indices de sensibilité peuvent être calculés en utilisant des techniques de régression linéaire. Ces indices fournissent une information sur l'influence de chaque variable d'entrée sur l'évaluation du modèle. Ils fournissent des informations sur le poids de chaque variable et permettent également de trier ces variables entre elles.

Une première méthode est d'évaluer les coefficients de corrélation linéaire (aussi appelés coefficients de Pearson) $\rho(.,.)$ entre une variable d'entrée X_j, et la variable de sortie Y :

$$\rho_j = \rho(X_j, Y) = \frac{Cov(X_j, Y)}{\sqrt{Var(X_j)Var(Y)}}. \tag{1.55}$$

Cette méthode, simple et nécessitant peu d'évaluations ($N \geq D + 1$), ne possède de sens que dans le cas proche d'une relation affine.

Lorsque le modèle est monotone (i.e. soit croissant, soit décroissant sur son domaine de définition), le coefficient de corrélation linéaire ne retranscrit pas précisément la relation entre les variables. Dans

ce cas, une méthode de calcul des coefficients de corrélation basés sur les rangs, le coefficient de Spearman, peut être appliquée. L'idée est de transformer une relation monotone entre les variables en une relation linéaire entre les rangs de ces variables.

Le coefficient, noté $\rho^s(.,.)$, est basé sur le rang des variables d'entrée et sur celui de la variable de sortie. Le vecteur $R_X = (R_{X_1}, \ldots, R_{X_D})$ est défini comme le vecteur des rangs des variables d'entrée et le vecteur R_Y, celui des rangs de la sortie. L'équation (1.56) donne la formule du calcul des coefficients :

$$\rho_j^s = \rho^s(X_j, Y) = \rho(R_{X_j}, R_Y). \tag{1.56}$$

Ces deux méthodes de régression linéaire ne sont pas exhaustives, il existe d'autres méthodes telles que les coefficients de régression standard ou les coefficients de régression partielle, plus coûteuses en calculs. Ces méthodes sont détaillées dans [Faivre et collab., 2013].

1.6 Méthodes pas-à-pas (*One-At-Time*)

Ce sont les méthodes simples et communes. À la manière d'un calcul dérivatif, elles sont basées sur la modification d'une entrée, une variable à la fois (i.e. *One-at-time* (OAT)), les autres restant fixes. Leur application est faite localement, avec un faible écart-type, sans parcours de l'espace de variation. Elles ne sont robustes que lorsque le modèle est linéaire. Ce processus générique est repris par de nombreux travaux, notamment dans la méthode *response surface design* [Box, 1952; Downing et collab., 1985; Montgomery, 2008; Myers et collab., 2016], dans les travaux de Cheng *et al.* [Cheng et collab., 1991].

Malgré le succès dû à leur simplicité [Saltelli et Annoni, 2010], Saltelli *et al.* déconseillent leur utilisation. Ils relèvent le fait que ces méthodes réalisent une analyse locale et qu'elles ne peuvent pas détecter les interactions entre variables. Lorsque la propriété de linéarité du modèle n'est pas connue, les auteurs préconisent d'utiliser d'autres méthodes, notamment celle des effets élémentaires (cf 1.7). De plus, dans [Saltelli et collab., 2008], les auteurs soulignent une autre limite liée au ratio entre le nombre de variables ayant un effet (ou influence) significatif sur le modèle et le nombre total de variables constituant la donnée en entrée. Il est en effet très courant d'étudier un modèle dont peu de variables ont une influence significative sur le modèle ; une étude locale aura fortement tendance à ne pas déceler cette information. Cela dépend en effet de la répartition de l'échantillon sur l'espace de recherche ainsi que des valeurs choisies pour les variables fixées.

1.7 Méthode des Effets Élémentaires (la méthode de Morris)

Morris a développé la méthode des effets élémentaires [Morris, 1991] permettant de détecter l'influence des variables d'entrée sur l'évaluation d'un modèle. La méthode de Morris permet de détecter l'influence des variables d'un modèle linéaire ou non linéaire, ainsi que les interactions entre les variables. C'est une méthode de criblage par discrétisation de l'espace de recherche. Cela signifie que l'espace de recherche est discrétisé selon une grille régulière.

La conception de la méthode est basée sur la répétition (de 5 à 10 fois) d'un plan OAT. À chaque étape, la méthode décale une variable X_j^i d'une certaine valeur (Δ_j), les autres restant fixes. $\Delta_j = U(-1,1)\ \delta$, $\delta = \frac{Q}{2(Q-1)}$, où Q est le nombre de niveaux de discrétisation de l'espace de recherche et k, le pas de déplacement. La méthode calcule ensuite l'effet élémentaire $EE_j(X^i)$ de ce décalage sur le modèle, selon l'équation (1.57) :

$$EE_j(X^i) \ = \ \frac{f(X_1^i, \ldots, X_j^i + \Delta_j, \ldots, X_D^i) - f(X^i)}{\Delta_j}. \tag{1.57}$$

La méthode trace alors une trajectoire pour un point X^i, illustrée par la figure 1.13. L'effet élémentaire décrit par l'équation (1.57) donne une évaluation locale de la sensibilité (j^{ieme} variable du i^{ieme} point). L'analyse globale est donnée par l'ensemble des trajectoires d'un nombre r de points $(X^i, i \in 1, ..., r)$ choisis aléatoirement dans l'espace de recherche.

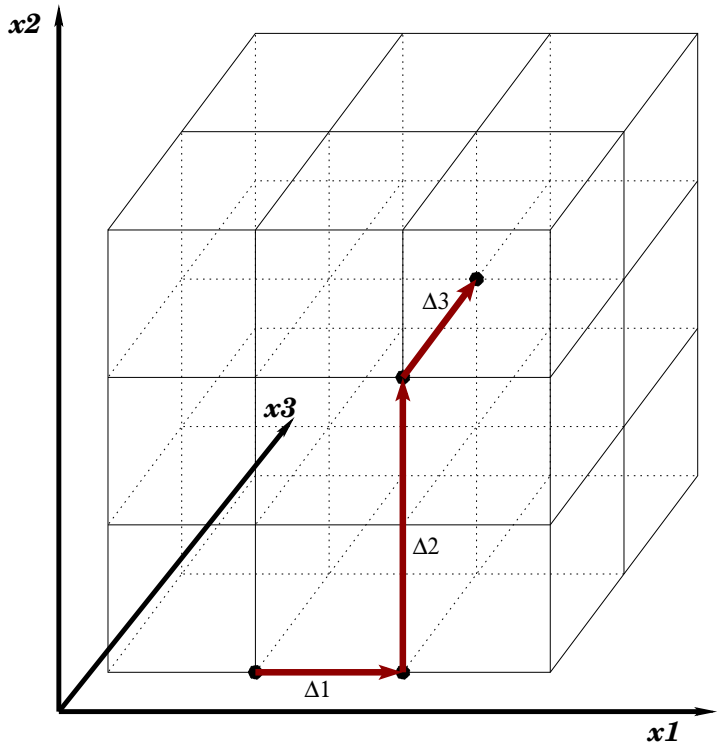

FIGURE 1.13 : Morris OAT : trajectoire d'un point

La sensibilité de chaque variable est ensuite évaluée par la moyenne des effets élémentaires, selon l'équation (1.58) :

$$\mu_j = \frac{1}{r} \sum_{i=1}^{r} EE_j(X^i). \tag{1.58}$$

Une faible valeur pour μ_j signifie que la j^{ieme} variable possède un faible effet sur le modèle, une forte valeur indique un fort effet sur le modèle. Néanmoins cette information n'est pas suffisante. Même si elle est un bon indicateur, en considérant seulement la moyenne, on pourrait omettre des informations sur un comportement non linéaire d'une ou plusieurs variables. L'étude inclut donc le calcul de l'écart-type des effets élémentaires, selon (1.59) :

$$\sigma_j = \sqrt{\frac{1}{r} \sum_{i=1}^{r} (EE_j(X^i) - \mu_j)^2}. \tag{1.59}$$

L'écart-type σ_j apporte plus d'information sur les effets d'une variable. Considérant les valeurs du couple moyenne absolue/écart-type (μ^*, σ), quatre cas de figure se présentent (cf également illustrations Fig. 2.1 et 3.1) :

- deux faibles valeurs de moyenne et d'écart-type impliquent que la variable n'a *pas ou peu d'effet* sur le modèle ;
- une faible valeur d'écart-type et une forte de moyenne impliquent que la variable possède un *fort effet linéaire* sur le modèle ;
- une faible valeur de moyenne et une forte d'écart-type impliquent que la variable possède un *fort effet non linéaire* sur le modèle ;
- deux fortes valeurs de moyenne et d'écart-type décrivent une variable possédant un *fort effet non linéaire* ou une *forte interaction* avec d'autres variables.

Dans [Campolongo et collab., 2007], Campolongo *et al.* suggèrent d'utiliser la moyenne absolue des effets élémentaires (1.60) en tant que mesure de l'influence d'une variable. Ils considèrent en effet que μ^* fournit une bonne approximation de l'indice total de sensibilité défini par Saltelli *et al.* dans [Saltelli, 2002b] et décrit par (1.60).

$$\mu_j^* = \frac{1}{r} \sum_{i=1}^{r} |EE_j(X^i)|. \tag{1.60}$$

Le processus complet est détaillé dans l'algorithme 1.8.

Cette méthode dite la méthode de Morris possède un faible coût calculatoire, permet de détecter des effets linéaires et non linéaires ainsi que des interactions entre variables. Néanmoins, cette méthode est contrainte par son plan d'expérience. En effet, elle ne calcule pas les effets élémentaires sur un ensemble aléatoire de points. Elle impose un ensemble de points de départ, puis effectue une série

Algorithme 1.8 : Algorithme de la méthode de Morris.

Initialiser r points aléatoires et leur évaluation
pour $j \leftarrow 1$ *à* D **faire**
  pour $i \leftarrow 1$ *à* r **faire**
    $\Delta \leftarrow random()$
    $X^{i\prime} \leftarrow (X_1^i, \ldots, X_j^i + \Delta, \ldots, X_D^i)$
    $EE_j(X^i) \leftarrow \frac{f(X^{i\prime}) - f(X^i)}{\Delta}$
    $X^i \leftarrow X^{i\prime}$
  fin
  Calculer μ_j^* et σ_j, selon (1.60) et (1.59)
fin

de déplacements unidimensionnels sur chacun de ces points. De plus, elle permet de définir si une variable possède ou non de l'influence sur le modèle, mais n'offre pas de méthode de mesure pour la quantifier. À noter que d'autres méthodes s'inspirent de la construction de trajectoires définissant les effets élémentaires, comme la méthode des escaliers en colimaçon [Jansen, 1999].

1.8 Méthodes basées sur l'analyse de la variance

Contrairement à la méthode de Morris, les méthodes basées sur l'analyse de la variance fournissent une mesure afin de quantifier et d'ordonner les influences des variables d'entrée. Les résultats d'une telle mesure sont les indices de sensibilité. C'est une méthode d'analyse globale, dans le sens où l'on considère la variabilité de la réponse en faisant varier les entrées sur l'ensemble de leur espace de variation. Comme introduit précédemment, deux mesures de sensibilité basées sur la variance sont utilisées ; l'indice principal (ou de premier ordre) et l'indice total ce sont les indices de sensibilité.

Les méthodes d'analyse de la variance ont en commun l'estimation de ces indices. C'est une opération extrêmement coûteuse en taille d'échantillon et en nombre d'évaluations. Ce sont en revanche les méthodes qui évaluent l'influence d'une variable avec la meilleure précision.

1.8.1 Les indices de Sobol

La méthode des indices de Sobol [Saltelli, 2002b] constitue la méthode la plus attractive car elle permet de quantifier de manière précise l'effet d'une variable sur la variance de Y. La variance de la réponse est décomposée en une somme de termes : $Var(Y) = V_{X_1} + V_{X_2} + \ldots + V_{X_k\ X_g} + V_{X_k\ X_p} + \ldots$ Chaque terme représente la variabilité de la réponse due à une variable ou à une interaction de plusieurs variables.

L'indice principal, pour la i^{ieme} variable, est donné par :

$$S_i = \frac{Var\left[\mathbb{E}(Y|X_i)\right]}{Var(Y)}. \tag{1.61}$$

Chaque terme donne le pourcentage de la variance de la réponse, attribuable soit à une variable soit à une interaction entre deux, trois ou plusieurs variables.

L'indice de sensibilité total est défini en équation (1.62) :

$$S_{T_i} = \frac{\mathbb{E}\left[Var(Y|X_j,\ j \neq i)\right]}{Var(Y)}. \tag{1.62}$$

Cet indice représente un pourcentage de variance de tous les effets imputables à la variable X_i, seule ou avec interactions, par rapport à celle de la solution.

Pour illustrer par un exemple, pour la fonction Ishigami de dimension $D = 3$:

$$f(X) = \sin(X_1) + 7\sin^2(X_2) + 0,1\ \ \sin(X_1)\sin^4(X_3),$$

où les X_i sont uniformément distribués sur $[-\pi, \pi]$.
Le calcul des indices de Sobol amène à la conclusion que $31,4\%$ de la variance de la réponse sont dus à la première variable X_1 ; $44,2\%$ sont dus à la seconde variable X_2 et $24,4\%$ sont expliqués par une interaction entre X_1 et X_3.

L'indice de Sobol de sensibilité total de X_i, S_{T_i}, est la somme des termes de la décomposition impliquant X_i. Il mesure la contribution de la variance de la réponse de X_i et ses interactions avec les autres variables. Considérant l'exemple précédent de la fonction Ishigami, l'indice total de la variable S_{T_1} est $55,8\%$, celui de S_{T_2} est $44,2\%$ et celui de S_{T_3} est $24,4\%$. La somme des effets totaux est supérieure à 100%, nous définissons alors un poids de variable p_i, selon (1.63) :

$$p_i = S_{T_i}/ST \tag{1.63}$$

où S_{T_i} est l'indice total de Sobol de la variable X_i et ST est la somme des indices totaux.

La figure 1.14 illustre la distribution des indices totaux de Sobol et des poids des variables.

Plusieurs méthodes pour estimer ces indices ont été développées, basées sur un échantillonnage par la méthode de Monte-Carlo [Saltelli, 2002a; Sobol', 2001, 1990]. Néanmoins, elles requièrent un nombre élevé d'évaluations pour réaliser une estimation précise des indices par la méthode de Monte-Carlo (Iooss et Mahévas [Faivre et collab., 2013] font état de l'ordre de 10000 évaluations pour une précision de 10%, pour une variable).

Cette méthode est applicable à des fonctions non linéaires, elle serait donc appropriée pour traiter

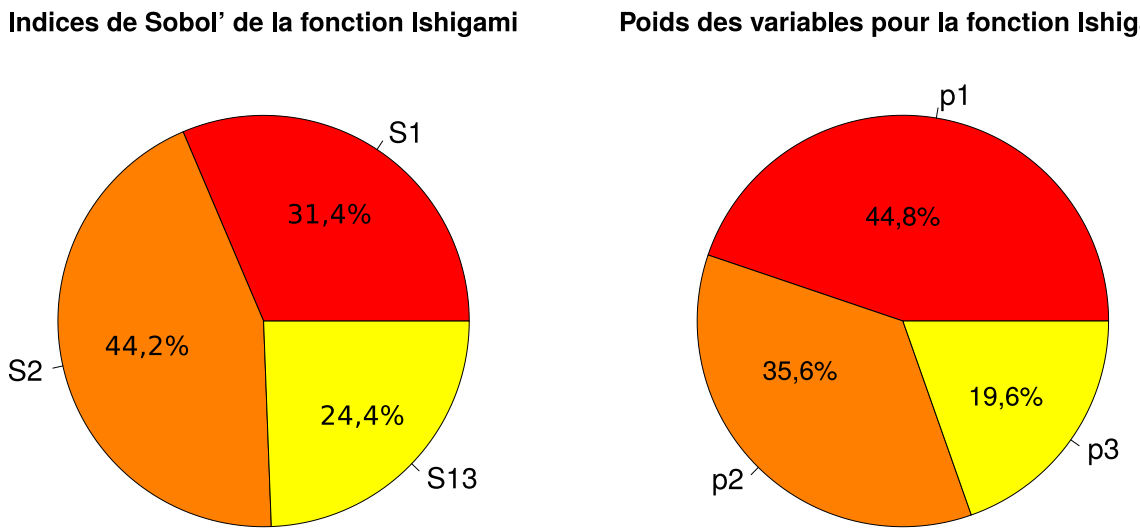

FIGURE 1.14 : Conversion des indices totaux de Sobol en poids.

des fonctions objectifs de problèmes d'optimisation non linéaires, mais le nombre trop important d'évaluations nécessaires la rend difficilement exploitable.

Des méthodes permettent de réduire le nombre d'évaluations afin de déterminer les indices, comme la méthode FAST ou l'utilisation d'un métamodèle.

1.8.2 La méthode FAST

Afin d'améliorer l'estimation des indices de sensibilité en termes de coût, Cuckier *et al.* ont développé une méthode basée sur une transformée de Fourier du modèle, la méthode FAST (*Fourier Amplitude Sensitivity Test*) [Cukier et collab., 1978]. Elle permet de déterminer les indices principaux des variables d'un modèle. L'échantillonnage est composé ici de trajectoires parcourant l'espace de variation de l'entrée, avec différentes fréquences. Une trajectoire est un ensemble de valeurs, pour une variable, généré par une fonction sinusoïdale et le vecteur de fréquences associé. La méthode a été étendue pour les indices de sensibilité totaux par Saltelli *et al.* [Saltelli et collab., 1999]. Cette méthode est moins coûteuse que la méthode par estimation de Monte-Carlo mais nécessite tout de même un nombre conséquent d'évaluations. Elle est moins robuste lorsque le nombre de dimensions augmente et est contrainte par une certaine régularité du modèle [Tissot et Prieur, 2012].

1.8.3 Méthode basée sur la construction de métamodèle

Un métamodèle peut être vu comme une simple fonction $Y = f(X)$. C'est une approximation d'un code de calcul lorsque celui-ci ne possède pas d'expression analytique. L'utilisation de métamodèles intervient lorsque le modèle étudié ne peut s'exprimer sous forme analytique ou lorsque le calcul de son évaluation est excessivement coûteux, a fortiori si l'analyse nécessite un nombre élevé d'évaluations. L'objectif est donc de créer une inférence du modèle par un modèle simplifié, en évaluant un nombre

restreint d'entrées.

Le choix des entrées, par répartition uniforme sur l'espace de recherche (*maximin design*, hypercube latin, etc.), est un problème crucial pour une estimation correcte. Le résultat fourni par le métamodèle sera une estimation de l'évaluation permettant d'approcher les indices de sensibilité. En effet, les indices de sensibilité sont évalués pour le métamodèle et non pour le modèle d'origine. Une correction doit alors être apportée, pour cela on évalue un critère d'ajustement. Iooss [Iooss et collab., 2010] souligne le fait qu'un métamodèle nécessite une validation en estimant une erreur quadratique ou absolue sur une base de tests.

Un métamodèle est construit selon un ensemble de techniques basées sur la régression et les moindres carrés. Il est décrit dans les travaux de Storlie et Helton [Storlie et Helton, 2008], Simpson *et al.* [Simpson et collab., 2001] et de Villa-Vialaneix *et al.* [Villa-Vialaneix et collab., 2012]. D'autres méthodes permettent d'approcher une telle surface (processus gaussien, SVM, réseau de neurones, etc.) [Bishop, 1995; Fang et collab., 2006; Santner et collab., 2003]. Dans [Iooss, 2011], Iooss indique que la formulation de certains métamodèles fournit une expression analytique des indices de sensibilité, (Chaos polynomial [Sudret, 2008], modèle linéaire complexe [Jourdan, 2012]) évitant l'utilisation d'une coûteuse estimation par la méthode de Monte-Carlo. Le cas échéant, une analyse des indices de Sobol, ou autre méthode, peut être effectuée sur le modèle de surface généré.

1.9 Conclusion

En premier lieu, dans ce chapitre, nous avons introduit les problèmes d'optimisation difficile et la manière dont ils peuvent être résolus de façon plus ou moins générique et efficiente par les métaheuristiques. Nous avons distingué les problèmes d'optimisation à variables discrètes et continues, les métaheuristiques mono-agent et à population, ainsi que les familles de métaphores évolutionnaires et de l'intelligence en essaim. Nous avons défini les stratégies d'exploration et d'exploitation qu'utilisent ces métaheuristiques pour diversifier la population ou intensifier la recherche au voisinage d'une solution prometteuse.

Parmi les algorithmes existants, nous avons étudié plus en détail les métaheuristiques d'évolution différentielle (DE), d'optimisation par essaim particulaire (PSO) et de colonie d'abeilles artificielles (ABC). Nous avons présenté les différentes améliorations proposées par étude de leurs caractéristiques intrinsèques.

Le lecteur pourra découvrir de nombreuses méthodes, non abordées dans ce travail, basées sur des métaphores (chauves-souris, termites, cafards, araignées, etc.). Néanmoins, loin de se restreindre à cette contrainte, de nombreuses stratégies de recherche et métaheuristiques ne respectent pas une figure de style quelconque et fournissent des résultats compétitifs. Nous pouvons citer la méthode

de Monte-Carlo [Hastings, 1970], le *branch and bound* stochastique [Norkin et collab., 1998], les métaheuristiques à population MLSDO et CMADO de Lepagnot *et al.* [Lepagnot, 2011], pour les problèmes d'optimisation dynamique ; MTS [Tseng et Chen, 2008], CUS et DEUS de Gardeux *et al.*[Gardeux et collab., 2009] pour l'optimisation continue en grande dimension. À ce titre, Sörensen, dans [Sörensen, 2015], se livre à une critique de la course à la métaphore dans l'élaboration de nouvelles méthodes. Le point clé pour définir ou améliorer un algorithme est d'identifier et d'utiliser les diverses méthodes et stratégies impliquées dans les métaheuristiques [Taillard, 2002] : la représentation du problème, la définition d'un voisinage, les opérations de génération d'une nouvelle solution, les schémas de diversification et d'intensification, l'utilisation d'une mémoire, etc.

Dans un second temps, nous avons présenté le principe de l'analyse de sensibilité, ainsi que les principales méthodes utilisées.

Dans le contexte de l'optimisation continue, le modèle correspond à la fonction objectif f, une entrée est un vecteur solution $X = (x_1, ..., x_n)$, un paramètre x_i est une dimension de la solution et la réponse est l'évaluation de la solution par la fonction objectif, $f(X)$. La fonction objectif est traitée comme une boîte noire, nous ne supposons donc aucune information sur ses caractéristiques de dérivabilité et de linéarité.

L'analyse de sensibilité a précédemment été appliquée à des données non issues de plan d'expérience ou d'échantillonnage statistique [Plischke et collab., 2013], ou encore dans le contexte de l'optimisation [Takahashi et collab., 2001], fournies par une métaheuristique [Avila et collab., 2006]. Par contre, elle n'a pas été utilisée jusqu'ici dans le but d'améliorer la convergence d'une métaheuristique, en acquérant de la connaissance sur le comportement d'une fonction.

Le rôle de l'analyse de sensibilité est ici de guider une recherche aléatoire afin de lui permettre de se concentrer sur les dimensions les plus prometteuses. Inclure une analyse de sensibilité dans une métaheuristique pourra aider à prioriser la direction de l'exploration ou de l'exploitation le long des dimensions d'intérêt. L'utilisation d'une méthode d'analyse de sensibilité dans le cadre d'une métaheuristique induit quelques contraintes orientant son choix. Les entrées sont fournies par la métaheuristique et non basées sur une répartition probabiliste et/ou plan d'expérience. Enfin elle ne doit pas nécessiter un grand nombre de points et d'évaluations.

Nous allons présenter deux types d'implémentations liées aux stratégies de déplacement utilisées dans différentes métaheuristiques. En premier lieu, nous proposons dans le chapitre 2 une amélioration de l'algorithme ABC, au déplacement unidimensionnel, par l'intégration de la méthode de Morris. Puis, au chapitre 3, nous allons généraliser l'approche aux métaheuristiques à déplacement multi-dimensionnel, en développant une nouvelle méthode d'analyse de sensibilité.

L'objectif commun est de permettre à une métaheuristique d'acquérir de la connaissance sur

l'incertitude de sa fonction d'évaluation, afin d'améliorer sa convergence en utilisant uniquement les évaluations effectuées au cours de son déroulement.

Chapitre 2

Élaboration de l'algorithme ABC-Morris pour un parcours guidé de l'espace de recherche

Ce chapitre présente ABC-Morris, une première intégration d'une méthode d'analyse de sensibilité, la méthode de Morris, dans plusieurs métaheuristiques de type colonie d'abeilles artificielles. Un ensemble de comparaisons avec les algorithmes de la littérature ainsi qu'un comparatif entre version originale et modifiée sont réalisés sur le benchmark *CEC 2013.*

Notions abordées dans ce chapitre :

- *Colonie d'abeilles artificielles ;*
- *Méthode de Morris ;*
- *Pourcentage de variables actives ;*
- *Influence d'une dimension.*

2.1 Introduction

Comme nous l'avons vu précédemment au chapitre 1, les algorithmes à population, basés sur l'intelligence en essaim ou les interactions sociales, ou les algorithmes évolutionnaires, ont provoqué un fort intérêt de recherche pour la résolution de problèmes d'optimisation. Certains sont basés sur la sélection des meilleurs individus, tandis que d'autres s'intéressent aux comportements d'une population peu intelligente mais extrêmement communicante, qui va échanger de l'information et collaborer. Ces comportements ont été décrits à travers les métaheuristiques de colonie de fourmis, d'essaim particulaire ou de colonie d'abeilles artificielles. Ce dernier algorithme, introduit par D. Karaboga en 2005 [Karaboga, 2005], décrit en 1.3.3, fait une analogie avec le comportement de différentes catégories d'abeilles afin d'explorer l'espace de recherche d'une fonction. Cet algorithme a prouvé son efficacité vis-à-vis des autres algorithmes à population, qu'ils soient de sélection (évolution différentielle) ou d'intelligence en essaim (PSO) [Karaboga et Basturk, 2008].

Les travaux autour de cet algorithme étudient le moyen de résoudre le problème dichotomique entre exploration de l'espace de recherche pour identifier les zones prometteuses et exploitation d'une zone prometteuse. Ils tentent aussi de solutionner le problème de convergence prématurée. Ces problématiques communes à toutes les métaheuristiques se voient solutionnées à travers différentes stratégies. Dans le cadre de l'algorithme ABC, ces stratégies sont les suivantes : l'amélioration de la recherche de voisinage, la quantification du caractère prometteur d'une solution (exploitation) et la recherche d'un nouvel individu (exploration). Ces questions ont été abordées en section 1.3.3.

Parmi les différentes stratégies étudiées, la question de l'influence de la dimension du problème à optimiser sur la fonction d'évaluation ne semble pas avoir été posée. Dans ce sens, l'analyse de sensibilité, comme présentée en section 1.4, permet de définir quelles variables (i.e. dimensions) d'un modèle (i.e. fonction) ont une influence significative sur celui-ci. Dans le contexte de l'optimisation continue, lorsque la fonction est connue, nous pouvons émettre l'hypothèse qu'il y a une faible incertitude quant à son résultat et que la plupart des influences des dimensions sont connues. Le rôle de l'analyse de sensibilité est alors de guider la recherche aléatoire d'une métaheuristique afin de se concentrer sur les dimensions les plus prometteuses. Intégrer une méthode d'analyse de sensibilité dans une métaheuristique permettrait ainsi de définir un tel comportement de recherche.

Ce chapitre présente comment une méthode d'analyse de sensibilité, telle que la méthode de Morris, peut s'intégrer aisément et de manière générique dans l'algorithme ABC et ses variantes. Elle permettra ainsi d'améliorer la qualité de sa recherche. L'algorithme ABC-Morris [Loubière et collab., 2016c], résultat de cette inclusion, est appliqué sur le *benchmark* CEC 2013 (*Conférence on Evolutionary Computation*, 21-23 Juin 2013, Cancun (Mexique)) afin d'étudier ses performances.

2.2 Description de notre variante de l'algorithme ABC

Nous présentons ici une version modifiée de l'algorithme ABC. L'idée est d'inclure dans cet algorithme la méthode de Morris afin de déterminer, tout au long de son exécution, les dimensions intéressantes à explorer. La génération d'une nouvelle solution par l'algorithme ABC est similaire à la construction des trajectoires de la méthode de Morris. Chacune analyse, dimension par dimension, l'influence d'un pas de décalage sur une variable du problème (i.e. le long d'une dimension).

La méthode de Morris ajuste l'influence des variables après chaque évaluation de la fonction objectif (cf équation (1.57)). Un taux d'influence (ou poids) est alors associé à chaque dimension. Ce taux permet de guider la sélection de dimension selon l'attractivité de cette dernière.

Durant les phases d'exploration et d'exploitation, l'algorithme ABC original engendre un nouvel individu en décalant aléatoirement une variable d'un individu considéré. Le décalage de la solution le long d'une dimension, défini dans l'équation (1.41), est similaire à ce qui est effectué par la méthode de Morris (cf équation (1.57)) afin de déterminer l'influence d'une dimension. Lors de la génération d'un nouvel individu par l'algorithme ABC, l'information apportée par le décalage effectué et l'évaluation obtenue vont être injectées dans le calcul des effets élémentaires. Tout au long de l'exécution de l'algorithme, nous pouvons utiliser les évaluations de la fonction objectif pour calculer de nouvelles valeurs de moyenne absolue μ^* (1.60) et d'écart-type σ (1.59), et ainsi, ajuster l'influence des dimensions dans une matrice d'effets élémentaires. Durant l'exécution de l'algorithme ABC, l'évaluation des influences de chaque dimension devient de plus en plus précise.

En effet, pour une solution X^i, une nouvelle solution candidate $X^{i'}$ est produite par l'équation (1.41). Cette solution est évaluée par la fonction objectif, $f(X^{i'})$. Si l'on considère $\Delta = \phi_{ij}(X^i_j - X^k_j)$ dans l'équation (1.41), nous pouvons aisément faire correspondre cette équation à celle des effets élémentaires (1.57) et donc appliquer la méthode de Morris. Contrairement à son principe, ici l'espace de recherche n'est pas discrétisé, les informations sont fournies par la métaheuristique : les trajectoires sont construites selon les déplacements des solutions composant la population et leurs évaluations, par l'appel de la fonction objectif.

L'objectif est de remplacer le tirage aléatoire d'une dimension à décaler j (équation (1.41)) par une sélection guidée de la dimension selon son influence sur la fonction à optimiser. Pour cela, nous devons prendre en compte les quatre cas décrits en 1.7, illustrés par la figure 2.1.

Nous avons vu, lors de sa description, que la méthode de Morris ne propose pas de mesure pour quantifier l'influence d'une variable. Néanmoins, nous pouvons noter que les variables ayant les influences les plus importantes sont celles dont le point formé par le couple (moyenne absolue, écart-type) est le plus éloigné de l'origine. La distance euclidienne est alors choisie comme fonction numérique d'évaluation de l'influence d'une dimension sur la fonction (2.1).

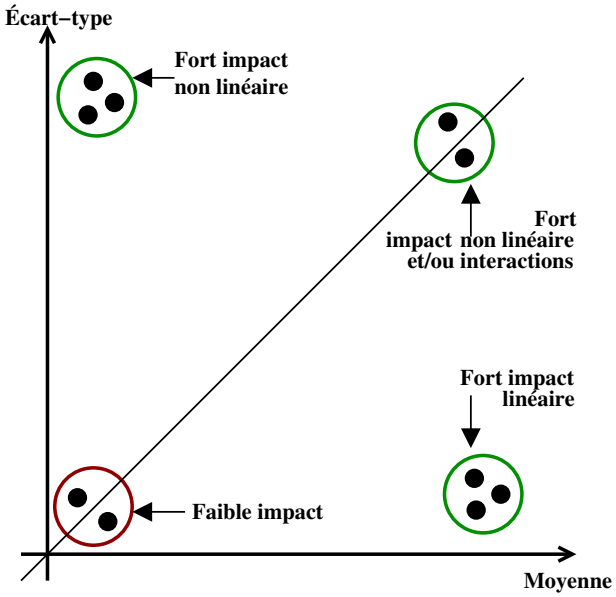

FIGURE 2.1 : Impact des dimensions selon le couple (moyenne, écart-type)

$$\forall\ j\ \in\ \{1,\ldots,D\},\ d(\mu_j^*,\sigma_j) = \sqrt{\mu_j^{*2}+\sigma_j^2}. \tag{2.1}$$

Le poids de chaque dimension est ensuite donné par le pourcentage d'influence de chaque dimension, selon (2.2) :

$$\forall\ j\ \in\ \{1,\ldots,D\},\ w_j\ =\ \frac{d(\mu_j^*,\sigma_j)}{\sum_{k=1}^{D} d(\mu_k^*,\sigma_k)}. \tag{2.2}$$

Le choix de la dimension à décaler ne sera plus une valeur aléatoire suivant une distribution uniforme dans $\{1,\ldots,D\}$, mais sera pris selon une loi de probabilité dont la distribution est définie par (2.2). Nous utilisons la méthode de simulation inverse pour remplacer la méthode de choix aléatoire d'une dimension à modifier. Cette méthode utilise un vecteur de pourcentages cumulés d'influence.

La sélection d'une dimension selon son poids est décrite dans l'algorithme 2.1.

À l'initialisation de la métaheuristique, une matrice d'effets élémentaires est créée (M_{EE}), comportant N lignes (N, nombre d'individus de la population initiale) et D colonnes (dimension du problème). Les valeurs de cette matrice sont initialisées à 1. Les moyennes absolues pour chaque dimension sont égales à 1 et les écarts-types à 0. Selon (2.1) et (2.2), les distances sont égales à 1 et les dimensions ont la même probabilité d'être sélectionnées : $1/D$.

Les algorithmes 2.1, 2.2 et 2.3 décrivent les modifications apportées à l'algorithme ABC afin d'intégrer la méthode de Morris.

La méthode de Morris maximise l'exploration de l'espace de recherche en testant de « petites » mais

Algorithme 2.1 : Algorithme de sélection de dimension

Fonction *selectDimensionPoids(w)* **:**
    i ← 1
    F ← w_1
    `// nombre aléatoire`
    rndm ← rand(0,1)
    `// // on itère les poids cumulés`
    tant que $F < rndm$ **faire**
        i ← i+1
        F ← F + w_i
    fin
    retourner i `// dimension sélectionnée`
fin

Algorithme 2.2 : Algorithme de génération d'individu : Morris-foraging

`//` X^i `une solution et` X^k`, un voisin choisi aléatoirement`
j ← selectDimensionPoids(w) (Alg. 2.1)
delta ← $\phi_{ij}(X^i_j - X^k_j)$
$X_j{}^{i'} = X^i_j +$ **delta**
X^i ← selectMeilleur(X^i, $X^{i'}$)
majEffetElementaire(M_{EE}, i, j, **delta**), (cf Eq. (1.57))

Algorithme 2.3 : Algorithme ABC-Morris

Initialiser la population de sources de nourriture
Initialiser la matrice des effets élémentaires
Initialiser les vecteurs μ^***,** σ **et** w
tant que *non critèreFin()* **faire**
    vol des ouvrières avec **Morris-foraging**, (cf Alg. 2.2)
    MAJ des vecteurs μ^***,** σ **et** w
    mise à jour des *fitness*
    vol des butineuses avec **Morris-foraging**, (cf Alg. 2.2)
    MAJ des vecteurs μ^***,** σ **et** w
    vol des exploratrices pour générer de nouvelles sources
    sauvegarder la meilleure solution
fin

aussi de « grandes » valeurs pour Δ. Dans notre algorithme, les valeurs pour Δ sont données par la différence $\Delta = \phi_{ij}(x_j^i - x_j^k)$. De par le caractère convergent de l'algorithme ABC, cette différence tend vers zéro. Par conséquent, $\Delta \equiv 0$ et donc $f(X + \Delta) \equiv f(X)$. Selon (1.57) et (1.58), deux cas de figure se présentent.

D'une part, une grande influence désigne une faible moyenne et un fort écart-type. Cela signifie qu'une valeur optimale pourrait se trouver dans l'intervalle $[X, X + \Delta]$, il y aurait alors un fort intérêt à explorer cette dimension. D'autre part, pour cette j^{ieme} dimension, tous les effets sont proches de zéro ; l'influence de cette dimension (μ^*) posséderait une faible influence et il n'y aurait que peu d'intérêt à explorer cette dimension, soit parce qu'elle n'aurait pas d'influence, soit parce que l'on aurait atteint une valeur optimale.

Ce comportement inhabituel de la méthode de Morris est induit par son utilisation dans le contexte d'une métaheuristique, et renforce son intérêt. Lorsque la métaheuristique a convergé pour la plupart des dimensions influentes, leurs poids deviendraient inférieurs, ou équivalents, à ceux des dimensions peu explorées. La métaheuristique pourrait alors explorer ces dimensions originellement moins influentes et se rapprocher ainsi de l'optimum global. L'avantage est donc que l'algorithme explorera en priorité les dimensions influentes, mais donnera une chance d'explorer les autres dimensions.

2.3 Tests et discussion

Afin d'évaluer l'influence de la méthode de Morris sur l'algorithme ABC et sur plusieurs de ces variantes récentes, nous utilisons le protocole et les fonctions de test de la compétition CEC 2013 [Li et collab., 2013]. L'étude réalisée se concentre sur deux aspects.

La première évaluation consiste à déterminer si l'intégration de la méthode de Morris dans l'algorithme ABC (ainsi que dans ses versions plus récentes) fournit des résultats équivalents ou meilleurs que leur version d'origine, lorsque l'ensemble des dimensions est actif. L'algorithme NBIPOP-aCMA-ES, meilleur algorithme de la compétition CEC 2013, est également inclus dans la comparaison, ainsi que les algorithmes PSO (dans sa version donnée en [Karaboga et Gorkemli, 2014]) et I-ABC [Li et collab., 2012]. Les résultats de ce test sont présentés dans le tableau 2.1.

La seconde évaluation teste l'efficacité d'adapter une méthode d'analyse de sensibilité pour une métaheuristique lorsque plusieurs dimensions n'ont pas, ou peu, d'impact sur la fonction objectif. Les versions standards de chaque algorithme sont comparées à leur version incluant la méthode de Morris. Les évaluations ont été réalisées sur le même *benchmark*, pour 75%, 50%, 25% et 10% de variables actives ; les résultats correspondants sont présentés dans les tableaux 2.5 à 2.8.

Des articles récents [Bast et Weber, 2005; Birattari et Dorigo, 2007; Derrac et collab., 2011] soulignent le fait que la moyenne et l'écart-type peuvent être inadaptés pour une comparaison sta-

tistique, du fait que la distribution des résultats ne peut pas être assimilée à une loi normale, mais est asymétrique avec une longue queue. En effet, même si un algorithme fournit de bonnes performances, un nombre réduit d'accidents statistiques (la fin de la queue) peut fausser considérablement la moyenne et l'écart-type. Les rangs statistiques ne sont pas affectés par les valeurs extrêmes. Pour cette raison, nous avons utilisé la médiane et le test de Wilcoxon (plutôt que le test usuel de Student), dans notre comparaison. En effet le test de Wilcoxon est basé sur les rangs alors que celui de Student est basé sur la moyenne. Un test statistique est utilisé pour déterminer si un échantillon conduit à l'acceptation ou à la réfutation de l'hypothèse nulle suivante, avec un niveau de confiance significatif :

$$\begin{cases} H_0 : \text{les deux algorithmes sont équivalents} \\ H_1 : \text{il y a une différence significative entre les algorithmes} \end{cases} \tag{2.3}$$

Le résultat du test de Wilcoxon est la p-valeur (Tableaux 2.2, 2.3, 2.4 et annexe A, tableau 4). Il est alors communément admis que si :

$$\begin{cases} p < 0,01 & : \text{il y a une très forte présomption contre } H_0 \\ p \in [0,01,\ 0,05[& : \text{il y a une forte présomption contre } H_0 \\ p \in [0,05,\ 0,1[& : \text{il y a une faible présomption contre } H_0 \\ p \geq 0,1 & : \text{il n'y a aucune présomption contre } H_0 \end{cases} \tag{2.4}$$

Le but est de déterminer s'il y a une différence significative entre la version standard de chaque algorithme et leur version incluant la méthode de Morris. Le comportement de la p-valeur sera observé selon l'évolution du pourcentage de dimensions actives.

2.3.1 Fonctions de test utilisées pour la validation

Le *benchmark* de la compétition CEC 2013 [Li et collab., 2013] est composé de 28 fonctions. Les fonctions $f1$ à $f5$ sont unimodales, les fonctions $f6$ à $f20$ de ce *benchmark* sont multimodales et les fonctions $f21$ à $f28$ sont des fonctions composées. Une rotation est appliquée aux fonctions, à l'exception des fonctions $f5$, $f11$, $f14$, $f17$ et $f22$.

2.3.2 Algorithmes utilisés

La méthode de Morris est testée sur l'algorithme ABC, mais aussi sur trois de ses variantes : GABC [Zhu et Kwong, 2010], MeABC [Bansal et collab., 2013] et qABC [Karaboga et Gorkemli, 2014]. Elle est incorporée dans chacune de ces métaheuristiques compatibles avec la contrainte définie de déplacement unidimensionnel.

En complément de l'algorithme ABC de base, ces variantes ont été retenues car elles possèdent

chacune une stratégie de recherche différente. GABC inclut, à la manière de PSO, la meilleure solution globale dans l'équation de génération d'un nouvel individu. MeABC effectue quant à lui une recherche mémétique afin d'améliorer l'exploitation autour de la meilleure solution. Enfin, qABC possède l'équation modifiée pour l'étape des butineuses : il inclut une recherche de voisinage afin de trouver localement le meilleur voisin à décaler.

2.3.3 Protocole de test

Pour une comparaison équitable, les différentes versions de l'algorithme ABC ont été implémentées selon l'algorithme $ABCimp1(FEs)$ donné par Mernik *et al.* [Mernik et collab., 2015].

Les règles sont celles de la compétition CEC 2013, le nombre maximum d'évaluations est fixé à 10^4 D, où D est la dimension, avec une valeur égale à 50. Pour l'ensemble des algorithmes, la taille de population est de 50 (i.e. 50 particules pour PSO, 25 ouvrières + 25 butineuses pour les différentes versions des algorithmes ABC). Le paramètre du nombre maximal pour le compteur de visites d'une solution est fixé à : $l = SN\ D$ pour les différentes versions des algorithmes ABC.

Ci-dessous, le détail du paramétrage particulier de chaque algorithme :

— **GABC**

La valeur $C = 1,5$ est tirée de la partie expérimentale de [Zhu et Kwong, 2010].

— **MeABC**

Les paramètres pour la phase *golden section search* sont $[a = -1,2,\ b = 1,2]$ et le *golden ratio*, $\psi = 0,618$, comme définis dans [Bansal et collab., 2013]. Le taux de perturbation (pr) est fixé à $pr = 0,4$, Pour la recherche locale, le critère de fin, $\epsilon = 0,01$. La constante C, héritée de l'algorithme GABC, est fixée à $C = 1,5$.

— **qABC**

Le rayon de voisinage $r = 1$, selon la partie expérimentale de [Karaboga et Gorkemli, 2014].

— **I-ABC**

Pour l'équation $w_{ij} = \phi_1 = \frac{1}{(1+exp(-fit(X^i)/ap)^{iter})}$, le paramètre ap est égal à la *fitness* de la meilleure source de nourriture après la phase d'initialisation.

— **PSO**

Pour l'équation de la vitesse, selon [Karaboga et Gorkemli, 2014] et [Karaboga et Akay, 2009], les deux facteurs cognitif et social sont fixés à $1,8$; le poids d'inertie est fixé à $0,6$.

2.3.4 Résultats et discussion

Le tableau 2.1 présente les résultats pour la première évaluation, les tableaux 2.5, 2.6, 2.7 et 2.8 présentent les résultats de la seconde évaluation. Chaque tableau correspond à un pourcentage

de dimensions actives. Ils contiennent les erreurs médianes pour chaque algorithme appliqué aux fonctions du *benchmark* CEC 2013 sur 51 exécutions. Les résultats en gras correspondent aux plus petites erreurs trouvées par un algorithme pour une fonction. Les cellules en gris indiquent que la version de l'algorithme contenant la méthode de Morris améliore la version originale. La colonne « Standard » désigne l'implémentation de l'algorithme telle que décrite dans son papier d'origine (cf liste en 2.3.3), la colonne « Morris » désigne la version contenant la méthode de Morris. Une version exhaustive de ces résultats (contenant aussi la moyenne, l'écart-type, le meilleur et le plus mauvais résultat) est fournie en annexe A.

2.3.5 Comparaison pour 100% de dimensions actives

Les algorithmes ABC, GABC, MeABC et qABC, dans leur version « Morris », sont comparés avec PSO, I-ABC et NBIPOP-aCMA-ES. PSO et I-ABC ont été inclus car ces algorithmes ne sont pas compatibles avec la méthode de Morris. PSO effectue un décalage sur toutes les dimensions, ce qui le rend inadéquat pour une méthode *One-At-Time*. Dans l'article [Li et collab., 2012], les auteurs définissent, pour l'algorithme I-ABC, l'équation de déplacement suivante : $v_i^j = x_i^j w_{ij} + 2(\phi_{ij} - 0,5)(x_i^j - x_k^j)\Phi_1 + \varphi_{ij}(x_{bsf}^j - x_k^j)\Phi_2$, qui ne peut pas être exprimée sous la forme $x_i^j = x_i^j + \Delta$. La méthode de Morris ne peut pas être utilisée dans ce cas non plus.

L'objectif est de confirmer que l'ajout de la méthode de Morris fournit de bons résultats face à d'autres stratégies de recherche locale.

TABLEAU 2.1 : Erreur médiane - 100% de dimensions actives - 50D

	NBIPOP-aCMA-ES	PSO	I-ABC	ABCMorris	GABCMorris	MeABCMorris	qABCMorris
f1	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**
f2	**0,00e+00**	4,91e+07	1,20e+07	2,75e+08	2,92e+08	2,49e+08	3,02e+08
f3	**0,00e+00**	1,96e+09	1,11e+09	2,16e+11	4,32e+12	9,41e+12	3,71e+11
f4	**0,00e+00**	6,12e+03	1,61e+05	1,59e+05	1,55e+05	1,57e+05	1,60e+05
f5	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**
f6	**0,00e+00**	4,91e+01	4,35e+01	4,35e+01	4,37e+01	4,41e+01	4,35e+01
f7	**3,65e+00**	5,87e+01	1,45e+02	1,04e+03	6,48e+02	1,12e+03	8,16e+02
f8	**2,11e+01**	**2,11e+01**	**2,11e+01**	**2,11e+01**	2,12e+01	**2,11e+01**	**2,11e+01**
f9	**7,06e+00**	5,25e+01	5,82e+01	6,07e+01	5,83e+01	5,71e+01	6,05e+01
f10	**0,00e+00**	7,98e+01	3,47e-01	5,01e-01	4,58e-01	4,11e-01	1,42e-01
f11	5,97e+00	2,16e+02	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**
f12	**4,97e+00**	3,17e+02	5,73e+02	8,13e+02	5,48e+02	5,69e+02	8,15e+02
f13	**6,96e+00**	3,54e+02	6,43e+02	8,08e+02	6,54e+02	6,38e+02	8,35e+02
f14	1,33e+03	7,80e+03	8,69e-01	8,77e-01	4,68e+00	5,31e+00	**1,88e-01**
f15	1,50e+03	5,80e+00	2,56e-03	3,68e-04	7,56e-05	4,42e-06	**2,80e-06**
f16	**4,40e-02**	2,78e+00	1,44e+00	1,37e+00	1,57e+00	1,68e+00	1,41e+00
f17	5,67e+01	2,70e+02	**5,08e+01**	5,08e+01	**5,08e+01**	**5,08e+01**	**5,08e+01**

Continue page suivante

TABLEAU 2.1 : Erreur médiane - 100% de dimensions actives - 50D

	NBIPOP-aCMA-ES	PSO	I-ABC	ABCMorris	GABCMorris	MeABCMorris	qABCMorris
f18	1,04e+02	2,70e+02	**5,02e+01**	**5,02e+01**	**5,02e+01**	**5,02e+01**	**5,02e+01**
f19	4,42e+00	2,01e+01	**6,87e-01**	1,08e+00	9,69e-01	1,14e+00	8,24e-01
f20	2,27e+01	**2,03e+01**	2,13e+01	2,17e+01	2,10e+01	2,11e+01	2,14e+01
f21	**2,00e+02**	4,00e+02	**2,00e+02**	2,02e+02	**2,00e+02**	**2,00e+02**	2,03e+02
f22	1,34e+03	9,45e+03	1,37e+01	2,36e+01	1,38e+01	1,30e+01	**9,80e+00**
f23	**1,49e+03**	1,28e+04	9,70e+03	1,02e+04	1,01e+04	9,59e+03	9,76e+03
f24	**2,45e+02**	3,31e+02	3,68e+02	3,78e+02	3,68e+02	3,68e+02	3,75e+02
f25	**2,49e+02**	3,92e+02	4,07e+02	4,22e+02	4,00e+02	4,01e+02	4,20e+02
f26	**2,00e+02**	2,34e+02	2,01e+02	2,02e+02	2,02e+02	2,02e+02	2,01e+02
f27	**7,77e+02**	1,73e+03	1,91e+03	1,98e+03	1,86e+03	1,86e+03	1,95e+03
f28	**4,00e+02**	4,00e+02	**4,00e+02**	**4,00e+02**	**4,00e+02**	**4,00e+02**	**4,00e+02**

À 100% de dimensions actives, considérant l'erreur médiane, l'algorithme NBIPOP-aCMA-ES fournit les meilleurs résultats pour la plupart des fonctions.

Parmi les quatre algorithmes ABC, l'association de la méthode de Morris à qABC est la plus performante. Ce dernier trouve la plus petite erreur médiane pour les fonctions $f11$, $f14$, $f17$ et $f22$. Ce sont des fonctions sur lesquelles aucune rotation n'est appliquée. qABCMorris fournit de meilleurs résultats que NBIPOP-aCMA-ES pour les fonctions $f15$, $f18$, $f19$ et $f20$. Les résultats pour les fonctions $f1$, $f5$, $f8$ et $f28$ sont du même ordre quel que soit l'algorithme. Pour les fonctions $f2$, $f3$ et $f4$, il y a clairement un problème de convergence, la méthode fournissant de moins bons résultats que PSO et I-ABC. Couplée avec GABC et MeABC, la méthode de Morris fournit de meilleurs résultats que qABCMorris, appliquée aux fonctions $f20$, $f21$, $f24$ et $f27$.

Inclure la méthode de Morris semble donc prometteur, dans le sens où elle permet de fournir des résultats équivalents ou meilleurs que l'algorithme NBIPOP-aCMA-ES sur douze des fonctions tests, notamment sur les fonctions auxquelles on n'applique aucune rotation.

L'intérêt supplémentaire d'utiliser une méthode d'analyse de sensibilité dans une métaheuristique est de permettre de détecter les dimensions non influentes et de réaliser ainsi une recherche guidée afin d'améliorer la convergence de l'algorithme.

2.3.5.1 Comparaison pour 75% à 10% de dimensions actives

Afin de souligner l'intérêt d'inclure une méthode d'analyse de sensibilité dans une métaheuristique, nous avons désactivé un sous-ensemble de dimensions des fonctions du *benchmark* CEC 2013. Nous avons donc créé quatre cas d'études supplémentaires comportant 75%, 50%, 25% et 10% de dimensions actives, les dimensions non actives sont exclues du calcul de l'évaluation de la fonction objectif. Nous comparons la version originale des algorithmes ABC, GABC, MeABC et qABC avec

TABLEAU 2.2 : Résultats du test de Wilcoxon pour les fonctions unimodales (f1-f5).

% de dimensions actives			ABC	GABC	MeABC	qABC
100%	Médiane	Standard	176976,8	150237,1	156243,2	177314,0
		Morris	159367,5	155410,4	156701,8	160357,2
	p-valeur		0,329	0,1442	0,1829	0,362
75%	Médiane	Standard	74408,95	55449,99	56618,62	66993,22
		Morris	57962,47	53344,88	54884,67	56291,46
	p-valeur		0,5516	0,3063	0,395	0,3906
50%	Médiane	Standard	24473,63	20972,17	22006,70	26030,55
		Morris	24565,20	23140,21	22981,29	24026,56
	p-valeur		0,4284	0,2361	0,2775	0,445
25%	Médiane	Standard	1910,559	1480,540	1730,046	2036,896
		Morris	1848,844	1977,862	1931,903	2211,583
	p-valeur		0,5019	0,3025	0,2156	0,4426
10%	Médiane	Standard	217,7777	101,7326	92,0726	214,5408
		Morris	118,5910	151,8928	164,9290	108,3281
	p-valeur		0,8991	0,2571	0,2012	0,6568

TABLEAU 2.3 : Résultats du test de Wilcoxon pour les fonctions multimodales (f6-f20).

% de dimensions actives			ABC	GABC	MeABC	qABC
100%	Médiane	Standard	25,72189	21,48108	21,53499	22,12543
		Morris	25,68912	34,02730	43,45319	22,05982
	p-valeur		0,4829	0,5969	0,382	0,4324
75%	Médiane	Standard	20,77774	20,92197	20,97797	15,67462
		Morris	16,06201	19,16711	20,95855	18,34306
	p-valeur		0,6159	0,7328	0,45	0,5614
50%	Médiane	Standard	15,581330	10,866522	11,298972	9,905582
		Morris	9,892052	8,845657	8,785226	14,735677
	p-valeur		0,8219	0,4805	0,4068	0,923
25%	Médiane	Standard	1,6608072	1,1504701	1,1060454	1,4687789
		Morris	0,9206584	0,4318588	0,4862368	0,6863805
	p-valeur		**2,281e-02**	**3,014e-03**	**8,872e-03**	**1,175e-02**
10%	Médiane	Standard	0,6673380	0,4646484	0,4304996	0,4352877
		Morris	0,3017337	0,2605361	0,2916821	0,2184999
	p-valeur		**1,613e-02**	0,3153	0,4267	**5,076e-03**

leur version « Morris », en étudiant le comportement de la convergence, alors que le nombre de dimensions actives diminue.

Les tableaux 2.2, 2.3 et 2.4 présentent les erreurs médianes et les p-valeurs obtenues pour un algorithme sur une instance de pourcentage de dimensions actives. Ils représentent les résultats appliqués aux trois typologies de fonctions décrites dans le *benchmark* : unimodales, multimodales et composées.

Pour les fonctions unimodales (Tableau 2.2), il n'y a pas de différence significative entre les deux versions de chaque algorithme. Ce résultat s'explique par le fait que, pour certaines fonctions, les deux versions de l'algorithme trouvent l'optimum, ou ne convergent pas.

La p-valeur, dans les tableaux 2.3 et 2.4, indique qu'il y a une différence significative entre les deux versions de l'algorithme en faveur de la version Morris pour 25% et 10% de variables actives. Il semble que la méthode de Morris améliore les algorithmes ABC et qABC.

La figure 2.2 illustre l'écart entre les deux méthodes. Elle représente l'erreur médiane pour toutes les fonctions, selon le pourcentage de dimensions actives. Cela confirme qu'en dessous de 50%, l'inclusion de la méthode de Morris est profitable aux algorithmes ABC et qABC. Au dessus de 50% de dimensions actives, les résultats sont équivalents.

TABLEAU 2.4 : Résultats du test de Wilcoxon pour les fonctions composées (f21-f28)

% de dimensions actives			ABC	GABC	MeABC	qABC
100%	Médiane	Standard	389,9754	378,9624	379,7882	389,7422
		Morris	394,5005	383,1154	378,9397	395,8599
	p-valeur		0,8527	0,9911	0,9687	0,6326
75%	Médiane	Standard	322,5105	309,1698	310,7032	318,9712
		Morris	321,3738	310,7043	311,0640	318,6720
	p-valeur		0,7865	0,9507	0,8487	0,7031
50%	Médiane	Standard	216,2855	232,3355	221,3151	229,2212
		Morris	222,6468	201,0732	201,0264	208,7197
	p-valeur		0,8752	0,149	0,3264	0,5247
25%	Médiane	Standard	102,5452	100,4211	100,3969	102,6109
		Morris	100,7050	100,1289	100,1528	100,9540
	p-valeur		**9,936e-05**	**1,82e-02**	**3,70e-02**	**4,41e-03**
10%	Médiane	Standard	103,481	100	100	105,1231
		Morris	100,000	100	100	100,0008
	p-valeur		**7,26e-03**	0,5506	0,8223	**1,81e-03**

Si l'on étudie, plus en détail par fonction, Tableau 4 (en annexe A), la faible p-valeur des lignes et colonnes « Total » indique une différence significative en faveur de la version Morris de chaque algorithme, à l'exception des fonctions $f2$, $f3$, $f7$ et $f15$. Pour ces fonctions, on retrouve la faible convergence des algorithmes ou bien le fait qu'ils trouvent des valeurs équivalentes.

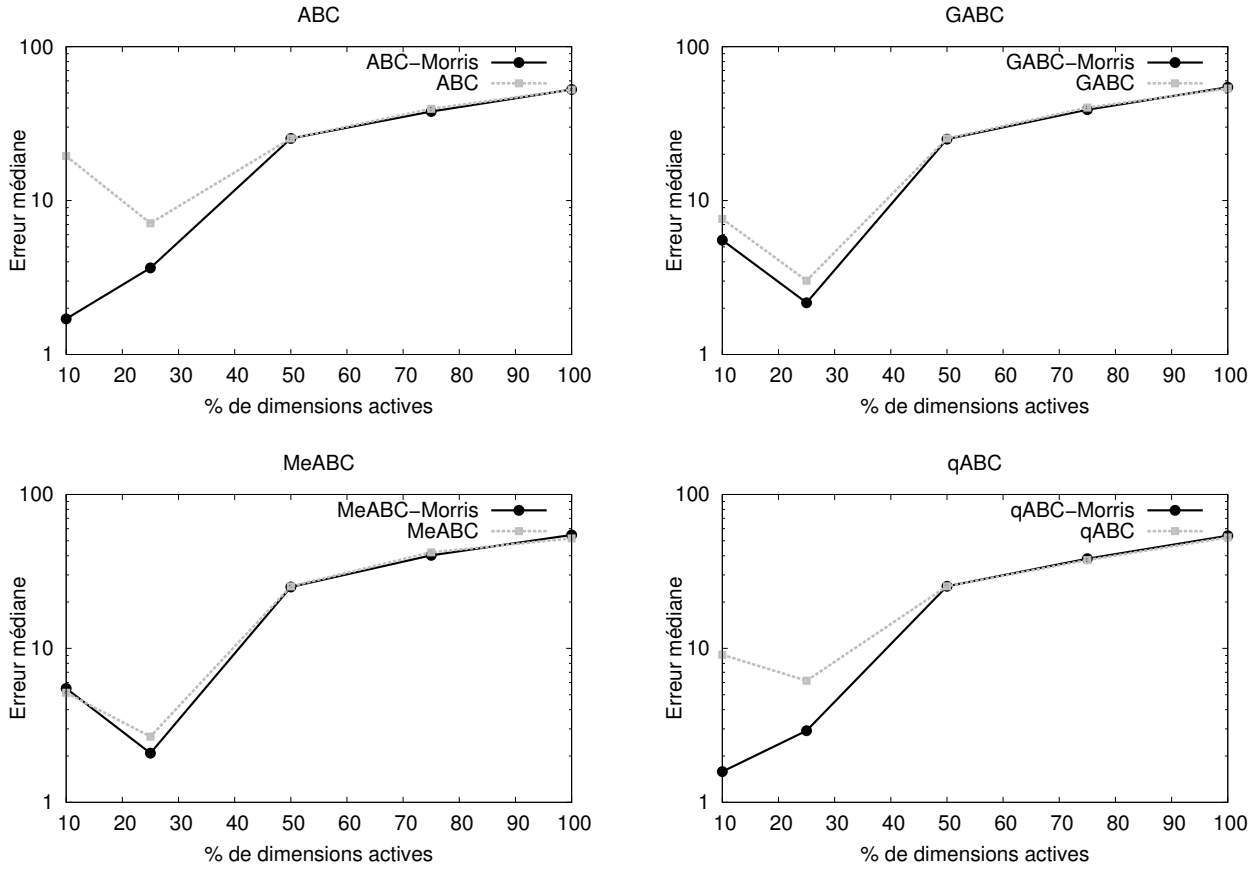

FIGURE 2.2 : Résultats globaux pour chaque algorithme, par pourcentage

Le tableau 4 apporte plus de détails pour chaque fonction. Les p-valeurs en gras correspondent à une différence significative en faveur de la version Morris d'un algorithme, une p-valeur en gris

représente une différence significative pour la version originale, enfin, une police d'écriture normale indique qu'il n'y a aucune différence significative entre les deux versions de l'algorithme.

Nous étudions alors ici les performances par pourcentage de dimensions actives.

Avec 75% de dimensions actives, les tableaux 2.2, 2.3 et 2.4 mettent en évidence que, globalement, il n'existe pas de différence significative entre les deux versions de l'algorithme. Néanmoins, si l'on se réfère aux erreurs médianes calculées pour ce pourcentage (cf Tableau 2.5), nous constatons que la méthode de Morris améliore l'algorithme d'origine pour au moins la moitié des fonctions. Elle permet d'obtenir la plus faible erreur médiane pour les fonctions $f1$, $f6$, $f10$, $f14$, $f16$ et $f22$, couplée avec qABC, tandis que GABCMorris l'obtient pour les fonctions $f4$, $f8$, $f9$, $f13$, $f23$ et $f25$. Aussi, selon le tableau 4, la méthode de Morris est fortement favorable à l'ensemble des algorithmes, sur les fonctions $f10$ et 14. Nous observons enfin que pour les fonctions $f2$ et $f3$, la méthode de Morris n'aide pas à la convergence et donne des résultats moins satisfaisants que l'algorithme original.

TABLEAU 2.5 : Erreur médiane - 75% de dimensions actives

	ABC		GABC		MeABC		qABC	
	Standard	Morris	Standard	Morris	Standard	Morris	Standard	Morris
f1	1,87e+01	1,37e+01	2,00e+01	1,65e+01	2,58e+01	1,30e+01	8,27e+00	**7,01e+00**
f2	3,83e+06	9,01e+07	3,40e+06	7,90e+07	4,46e+06	6,40e+07	**3,37e+06**	1,19e+08
f3	3,67e+09	1,32e+12	**2,91e+09**	5,62e+14	3,81e+09	7,81e+13	2,92e+09	6,52e+12
f4	7,44e+04	5,80e+04	5,55e+04	**5,33e+04**	5,66e+04	5,49e+04	6,70e+04	5,63e+04
f5	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**
f6	7,69e+00	5,93e+00	1,11e+01	9,58e+00	9,94e+00	9,26e+00	3,54e+00	**2,51e+00**
f7	1,12e+02	4,36e+03	1,01e+02	4,07e+03	**9,27e+01**	5,65e+03	1,23e+02	2,86e+03
f8	2,10e+01	2,10e+01	2,10e+01	**2,10e+01**	2,10e+01	2,10e+01	2,10e+01	2,10e+01
f9	4,28e+01	4,30e+01	4,09e+01	**4,04e+01**	4,15e+01	4,16e+01	4,26e+01	4,22e+01
f10	1,29e+01	1,04e+01	1,44e+01	1,08e+01	1,42e+01	1,18e+01	6,24e+00	**4,77e+00**
f11	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**
f12	2,04e+02	2,06e+02	**1,53e+02**	1,62e+02	1,59e+02	1,56e+02	2,00e+02	2,01e+02
f13	2,74e+02	2,73e+02	2,38e+02	**2,30e+02**	2,39e+02	2,39e+02	2,74e+02	2,68e+02
f14	2,39e-01	1,52e-01	2,04e-01	1,45e-01	2,17e-01	1,52e-01	2,03e-01	**1,44e-01**
f15	**0,00e+00**	7,01e-05	**0,00e+00**	7,31e-07	1,69e-06	7,48e-07	**0,00e+00**	2,35e-07
f16	1,25e+00	1,16e+00	1,51e+00	1,36e+00	1,34e+00	1,29e+00	1,21e+00	**1,16e+00**
f17	3,75e+01	**3,75e+01**	**3,75e+01**	**3,75e+01**	**3,75e+01**	**3,75e+01**	**3,75e+01**	**3,75e+01**
f18	**3,70e+01**	**3,70e+01**	**3,70e+01**	**3,70e+01**	**3,70e+01**	**3,70e+01**	**3,70e+01**	**3,70e+01**
f19	3,80e-01	6,10e-01	4,62e-01	4,20e-01	5,62e-01	4,07e-01	**3,41e-01**	4,99e-01
f20	1,50e+01	1,51e+01	**1,43e+01**	1,44e+01	1,46e+01	1,44e+01	1,51e+01	1,50e+01
f21	**2,00e+02**	**2,00e+02**	**2,00e+02**	**2,00e+02**	**2,00e+02**	**2,00e+02**	**2,00e+02**	**2,00e+02**
f22	2,05e+01	1,71e+01	1,56e+01	1,58e+01	3,49e+01	1,56e+01	1,53e+01	**1,50e+01**
f23	7,19e+03	6,88e+03	6,85e+03	**6,71e+03**	7,15e+03	6,87e+03	6,93e+03	7,05e+03
f24	3,17e+02	3,18e+02	**3,04e+02**	3,08e+02	3,06e+02	3,04e+02	3,12e+02	3,13e+02
f25	3,31e+02	3,29e+02	3,21e+02	**3,17e+02**	3,21e+02	3,18e+02	3,29e+02	3,27e+02
f26	2,01e+02	2,01e+02	2,01e+02	2,01e+02	2,01e+02	2,01e+02	**2,00e+02**	**2,00e+02**
f27	**4,00e+02**	**4,00e+02**	**4,00e+02**	**4,00e+02**	**4,00e+02**	**4,00e+02**	**4,00e+02**	**4,00e+02**

Continue page suivante

TABLEAU 2.5 : Erreur médiane - 75% de dimensions actives

	ABC		GABC		MeABC		qABC	
	Standard	Morris	Standard	Morris	Standard	Morris	Standard	Morris
f28	4,02e+02	4,03e+02	**4,00e+02**	**4,00e+02**	**4,00e+02**	**4,00e+02**	4,03e+02	4,03e+02

Une amélioration est observable pour 50% de dimensions actives (Tableau 2.6). qABCMorris fournit des résultats équivalents ou meilleurs sur la plupart des fonctions, à l'exception des fonctions $f2$, $f3$ et $f7$. L'algorithme obtient la plus petite erreur médiane pour les fonctions $f1$, $f6$ et $f10$. La méthode de Morris améliore aussi les algorithmes GABC et MeABC. GABCMorris fournit la plus petite erreur médiane pour les fonctions $f13$, $f19$, $f25$ et $f26$; MeABCMorris, pour les fonctions $f9$, $f12$ et $f24$.

TABLEAU 2.6 : Erreur médiane - 50% de dimensions actives

	ABC		GABC		MeABC		qABC	
	Standard	Morris	Standard	Morris	Standard	Morris	Standard	Morris
f1	3,92e+00	1,76e+00	4,06e+00	1,97e+00	3,04e+00	1,60e+00	1,98e+00	**8,55e-01**
f2	1,57e+06	2,46e+07	1,34e+06	1,89e+07	1,88e+06	1,60e+07	**1,32e+06**	1,99e+07
f3	2,41e+08	3,01e+10	**1,15e+08**	4,42e+10	1,72e+08	5,01e+10	2,11e+08	4,02e+10
f4	2,45e+04	2,46e+04	**2,10e+04**	2,31e+04	2,20e+04	2,30e+04	2,60e+04	2,40e+04
f5	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**
f6	3,52e+00	8,82e-01	4,58e+00	1,75e+00	3,65e+00	1,57e+00	6,07e-01	**2,70e-01**
f7	6,65e+01	1,06e+02	**5,33e+01**	1,33e+02	5,44e+01	1,18e+02	6,69e+01	9,63e+01
f8	2,09e+01	**2,08e+01**	2,09e+01	2,09e+01	2,09e+01	2,09e+01	2,09e+01	2,09e+01
f9	2,27e+01	2,26e+01	2,08e+01	2,06e+01	2,03e+01	**1,97e+01**	2,27e+01	2,19e+01
f10	4,36e+00	2,79e+00	5,45e+00	3,16e+00	4,95e+00	3,09e+00	2,13e+00	**1,67e+00**
f11	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**
f12	8,84e+01	9,35e+01	8,47e+01	7,64e+01	7,94e+01	**6,98e+01**	8,95e+01	7,95e+01
f13	1,21e+02	1,24e+02	1,10e+02	**1,04e+02**	1,13e+02	1,07e+02	1,28e+02	1,25e+02
f14	2,00e-01	1,25e-01	1,50e-01	**5,00e-02**	1,75e-01	**5,00e-02**	1,75e-01	8,75e-02
f15	**0,00e+00**	2,50e-05	1,19e-05	6,04e-08	4,05e-06	8,44e-08	**0,00e+00**	4,10e-08
f16	8,57e-01	**7,60e-01**	9,68e-01	8,00e-01	9,79e-01	8,19e-01	8,49e-01	8,15e-01
f17	**2,53e+01**	**2,53e+01**	**2,53e+01**	**2,53e+01**	**2,53e+01**	**2,53e+01**	**2,53e+01**	**2,53e+01**
f18	**2,51e+01**	**2,51e+01**	**2,51e+01**	**2,51e+01**	**2,51e+01**	**2,51e+01**	**2,51e+01**	**2,51e+01**
f19	1,89e-01	2,51e-01	2,20e-01	**1,58e-01**	2,36e-01	1,75e-01	1,81e-01	2,33e-01
f20	9,21e+00	9,27e+00	8,78e+00	**8,46e+00**	8,64e+00	8,60e+00	9,18e+00	8,93e+00
f21	1,02e+02	1,04e+02	**1,00e+02**	**1,00e+02**	**1,00e+02**	**1,00e+02**	1,02e+02	1,08e+02
f22	2,51e+01	2,03e+01	1,99e+01	**1,36e+01**	1,92e+01	**1,36e+01**	1,99e+01	1,46e+01
f23	3,85e+03	3,82e+03	3,89e+03	**3,76e+03**	3,98e+03	3,77e+03	4,09e+03	4,07e+03
f24	2,12e+02	2,04e+02	1,96e+02	1,88e+02	1,93e+02	**1,86e+02**	2,05e+02	2,00e+02
f25	2,56e+02	2,56e+02	2,51e+02	**2,46e+02**	2,52e+02	2,48e+02	2,57e+02	2,47e+02
f26	1,88e+02	1,80e+02	1,74e+02	**1,67e+02**	1,68e+02	1,69e+02	1,68e+02	1,72e+02
f27	**4,00e+02**	**4,00e+02**	**4,00e+02**	**4,00e+02**	**4,00e+02**	**4,00e+02**	**4,00e+02**	**4,00e+02**
f28	**3,00e+02**	**3,00e+02**	**3,00e+02**	**3,00e+02**	**3,00e+02**	**3,00e+02**	**3,00e+02**	**3,00e+02**

Les tableaux correspondant à 25% et 10% de dimensions actives (Tableaux 2.7 et 2.8) amènent aux mêmes conclusions, avec une efficacité renforcée alors que le nombre de dimensions actives diminue. À 25%, qABCMorris atteint la plus petite erreur médiane pour les fonctions $f1$, $f6$, $f10$ et $f12$, GABCMorris atteint cette plus petite erreur pour les fonctions $f13$, $f16$, $f17$, $f19$ et $f21$, et enfin MeMorris l'atteint pour les fonctions $f8$, $f8$, $f18$ et $f20$. La version « Morris » des algorithmes fournit des résultats meilleurs ou équivalents à la version d'origine pour presque toutes les fonctions.

TABLEAU 2.7 : Erreur médiane - 25% de dimensions actives

	ABC		GABC		MeABC		qABC	
	Standard	Morris	Standard	Morris	Standard	Morris	Standard	Morris
f1	1,56e-01	4,82e-02	1,65e-01	4,60e-02	1,11e-01	4,76e-02	1,31e-01	**3,76e-02**
f2	3,26e+05	6,69e+05	3,01e+05	7,68e+05	2,95e+05	7,37e+05	**2,84e+05**	8,02e+05
f3	4,82e+06	5,38e+08	**1,89e+06**	7,91e+08	2,02e+06	8,84e+08	5,45e+06	5,34e+08
f4	1,91e+03	1,85e+03	**1,48e+03**	1,98e+03	1,73e+03	1,93e+03	2,04e+03	2,21e+03
f5	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**
f6	9,77e-02	1,18e-02	8,72e-02	1,34e-02	5,71e-02	2,46e-02	4,91e-02	**4,53e-03**
f7	1,25e+01	1,46e+01	**7,99e+00**	9,58e+00	8,27e+00	8,93e+00	1,37e+01	1,25e+01
f8	1,87e+01	1,77e+01	1,46e+01	1,33e+01	1,19e+01	**1,14e+01**	1,84e+01	1,61e+01
f9	3,09e+00	3,20e+00	2,19e+00	2,09e+00	2,32e+00	**1,93e+00**	3,23e+00	2,92e+00
f10	1,65e+00	1,05e+00	1,71e+00	9,87e-01	1,85e+00	1,05e+00	1,71e+00	**9,61e-01**
f11	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**
f12	1,06e+00	2,95e-01	7,98e-01	2,43e-01	1,05e+00	3,42e-01	4,27e-01	**1,54e-01**
f13	1,21e+00	9,74e-01	9,78e-01	**2,97e-01**	7,27e-01	4,83e-01	1,21e+00	9,97e-01
f14	1,05e-01	**0,00e+00**	5,21e-02	**0,00e+00**	5,21e-02	**0,00e+00**	1,04e-01	**0,00e+00**
f15	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**
f16	4,86e-01	4,05e-01	4,91e-01	**3,87e-01**	5,81e-01	3,97e-01	4,81e-01	3,94e-01
f17	1,22e+01	1,22e+01	1,22e+01	**6,10e+00**	1,22e+01	1,22e+01	1,22e+01	1,22e+01
f18	1,26e+01	1,26e+01	1,26e+01	1,26e+01	8,62e+00	**0,00e+00**	1,26e+01	6,39e-01
f19	4,49e-02	4,45e-02	3,59e-02	**2,08e-02**	5,13e-02	2,83e-02	4,81e-02	4,83e-02
f20	2,72e+00	2,70e+00	2,37e+00	2,19e+00	2,39e+00	**2,09e+00**	2,69e+00	2,69e+00
f21	2,00e+02	1,05e+02	2,00e+02	**1,00e+02**	1,02e+02	1,08e+02	2,00e+02	1,09e+02
f22	6,78e+00	8,63e-02	4,05e+00	**0,00e+00**	2,47e-01	**0,00e+00**	5,89e+00	4,83e-04
f23	1,21e+03	1,07e+03	**9,21e+02**	9,28e+02	1,00e+03	9,88e+02	1,21e+03	1,04e+03
f24	1,02e+02	1,01e+02	1,01e+02	**1,01e+02**	1,01e+02	**1,01e+02**	1,02e+02	1,01e+02
f25	1,00e+02	**1,00e+02**	**1,00e+02**	**1,00e+02**	**1,00e+02**	**1,00e+02**	**1,00e+02**	**1,00e+02**
f26	1,01e+02	1,00e+02	1,00e+02	**1,00e+02**	1,00e+02	**1,00e+02**	1,01e+02	1,00e+02
f27	1,49e+02	3,76e+02	3,62e+02	3,64e+02	3,62e+02	3,54e+02	**1,16e+02**	3,83e+02
f28	1,02e+02	**1,00e+02**	**1,00e+02**	**1,00e+02**	**1,00e+02**	**1,00e+02**	1,09e+02	1,00e+02

Pour 10% de dimensions actives, nous pouvons établir le même constat, remarquant notamment que ABCMorris et qABCMorris fournissent de meilleurs résultats sur les fonctions composées.

Cela confirme les résultats observés des tableaux 2.2, 2.3, 2.4 et annexe A, tableau 4). Il existe une différence significative en faveur de la version « Morris » des algorithmes.

TABLEAU 2.8 : Erreur médiane - 10% de dimensions actives

	ABC		GABC		MeABC		qABC	
	Standard	Morris	Standard	Morris	Standard	Morris	Standard	Morris
f1	2,33e-02	**0,00e+00**	3,28e-03	8,73e-07	4,07e-03	1,38e-06	3,71e-02	1,63e-07
f2	1,79e+03	1,81e+03	1,31e+03	**1,11e+03**	1,40e+03	2,47e+03	1,84e+03	2,39e+03
f3	3,62e+03	1,99e+05	5,40e+02	3,00e+05	**4,56e+02**	4,28e+05	4,25e+03	8,43e+04
f4	6,40e+01	3,85e+01	3,35e+01	4,57e+01	3,67e+01	4,90e+01	8,70e+01	**3,34e+01**
f5	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**
f6	1,28e-02	7,34e-04	4,16e-03	2,08e-03	4,85e-03	1,66e-03	9,79e-03	**3,73e-04**
f7	2,33e-01	1,38e-01	7,96e-02	**3,77e-02**	7,43e-02	5,27e-02	2,63e-01	2,31e-01
f8	9,51e-01	2,50e-01	1,45e-01	4,80e-02	2,54e-01	**3,27e-02**	1,24e+00	1,68e-01
f9	1,85e-01	1,07e-01	1,11e-01	**6,63e-02**	1,08e-01	6,66e-02	2,17e-01	1,36e-01
f10	2,64e-01	1,71e-01	1,50e-01	1,05e-01	1,84e-01	**1,01e-01**	2,62e-01	1,81e-01
f11	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**
f12	7,62e-04	4,81e-04	2,90e-04	4,94e-04	3,27e-04	**2,73e-04**	7,27e-04	4,74e-04
f13	7,78e-04	4,95e-04	3,11e-04	1,99e-04	2,88e-04	**1,72e-04**	1,23e-03	3,29e-04
f14	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	1,53e-07	**0,00e+00**
f15	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**
f16	3,85e-01	3,02e-01	4,14e-01	**2,68e-01**	4,21e-01	2,97e-01	4,23e-01	2,83e-01
f17	1,81e+00	5,21e-08	6,35e-03	**0,00e+00**	1,93e-01	**0,00e+00**	2,16e+00	8,53e-02
f18	2,49e+00	6,90e-03	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	1,49e+00	2,89e-07
f19	1,65e-05	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	2,03e-04	3,36e-07
f20	2,37e-01	1,17e-01	1,34e-01	**7,12e-02**	1,22e-01	9,62e-02	1,97e-01	1,22e-01
f21	2,28e+00	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	5,13e+00	5,50e-02
f22	4,96e-04	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	5,43e-01	**0,00e+00**
f23	1,87e+02	1,22e+02	1,20e+02	1,25e+02	1,33e+02	**1,00e+02**	1,67e+02	1,32e+02
f24	4,04e+01	1,90e+01	2,64e+01	1,77e+01	2,71e+01	**1,56e+01**	3,01e+01	1,78e+01
f25	1,00e+02	1,00e+02	1,00e+02	1,00e+02	1,00e+02	1,00e+02	1,00e+02	**9,86e+01**
f26	1,27e+01	2,61e+00	1,01e+01	3,36e+00	4,12e+00	6,40e+00	8,76e+00	**2,47e+00**
f27	1,00e+02	**9,46e+01**	1,00e+02	1,00e+02	1,00e+02	1,00e+02	1,00e+02	1,00e+02
f28	1,00e+02	**1,07e+00**	7,69e+01	2,34e+00	3,70e+01	5,27e+01	1,01e+02	2,42e+00

Les résultats globaux pour chaque pourcentage de dimensions actives présentent une faible p-valeur (Tableaux 2.3, 2.4) lorsque le nombre de dimensions actives est faible vis-à-vis de la dimension totale du problème (pour 10% et 25%).

Néanmoins, la différence n'est pas toujours significative. En effet, dans la plupart des cas, les résultats sont proches entre les deux versions de chaque algorithme, à l'exception des fonctions $f2$ et $f3$, qui ne sont jamais améliorées, ou des fonctions $f14$, $f17$ et $f22$ qui le sont significativement.

2.3.6 La question de la rotation

Pour chaque instance de pourcentage, l'ajout de la méthode de Morris fournit de bons résultats sur les fonctions $f5$, $f11$, $f14$, $f17$ et $f22$. Ces fonctions sont celles du *benchmark* sur lesquelles aucune rotation n'est appliquée. Le *benchmark* CEC 2013 est élaboré de manière à ce que, pour la majorité des fonctions, une rotation soit effectuée sur le vecteur solution avant évaluation de la fonction. La méthode de Morris détecte l'influence de chaque dimension X_j du vecteur solution X, pour une fonction f. Lorsqu'une rotation $rot()$ est appliquée sur la solution X, l'évaluation de la fonction correspond à $f(rot(X))$. La méthode de Morris n'évalue donc plus l'influence du paramètre X_j mais celle du paramètre $rot_j(X)$.

Lorsque l'on désactive un sous-ensemble de dimensions, ce comportement détériore la qualité, voire la cohérence de l'évaluation de l'influence et de fait celle du processus de recherche de la métaheuristique.

Comme indiqué précédemment, dans le but de démontrer l'intérêt de la méthode, des tests supplémentaires ont été réalisés, désactivant les rotations sur l'ensemble des fonctions du *benchmark* CEC 2013. Les tableaux 2.9, 2.10, 2.11 présentent les résultats pour les algorithmes I-ABC, qABC et qABCMorris, pour respectivement 100%, 50% et 25% de dimensions actives, uniquement sur les fonctions modifiées par cette désactivation.

Les codes de présentation sont identiques aux tableaux précédents. En plus de la médiane, nous présentons la moyenne et l'écart-type.

TABLEAU 2.9 : Résultats pour 100% de dim. actives - Fonctions sans rotation

	I-ABC			qABC					
	Standard			Standard			Morris		
	Médiane	Moy.	Éc.-Type	Médiane	Moy.	Éc.-Type	Médiane	Moy.	Éc.-Type
f1	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**
f2	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**
f3	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	2,72e+17	9,49e+19	2,61e+20
f4	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**
f6	6,20e-03	4,88e-02	1,35e-01	**1,75e-03**	**5,10e-03**	**6,85e-03**	4,69e-03	1,71e-02	3,92e-02
f7	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	5,96e+05	1,26e+06	2,65e+06
f8	**2,00e+01**	**2,00e+01**	**1,86e-05**	**2,00e+01**	**2,00e+01**	**1,54e-05**	**2,00e+01**	**2,00e+01**	**2,83e-05**
f9	8,23e-05	1,05e-01	1,60e-01	**1,18e-05**	**3,03e-05**	**5,43e-05**	1,30e-03	1,47e-02	3,51e-02
f10	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**4,93e-04**	**2,70e-03**
f12	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**
f13	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**
f15	1,04e+00	1,53e+00	1,45e+00	**1,60e-01**	**2,17e-01**	**1,91e-01**	2,15e-01	3,80e-01	3,16e-01
f16	1,13e-03	1,10e-03	3,33e-04	**5,59e-04**	**5,91e-04**	**1,58e-04**	9,26e-04	9,02e-04	1,73e-04
f18	**5,08e+01**	**5,08e+01**	**6,11e-03**	**5,08e+01**	**5,08e+01**	**3,44e-03**	**5,08e+01**	**5,08e+01**	**1,14e-02**
f19	6,88e-01	7,19e-01	2,43e-01	**4,28e-01**	**4,74e-01**	**1,50e-01**	8,15e-01	1,23e+00	1,94e+00
f20	7,87e-01	8,03e-01	1,48e-01	**6,99e-01**	**7,56e-01**	**2,33e-01**	1,18e+00	1,29e+00	5,55e-01
f21	**2,00e+02**	**2,11e+02**	**3,84e+01**	**2,00e+02**	**1,99e+02**	**2,82e+01**	2,03e+02	2,22e+02	3,51e+01
f23	1,37e+01	1,89e+01	2,62e+01	**1,07e+01**	**1,14e+01**	**3,30e+00**	**1,07e+01**	**1,09e+01**	**2,49e+00**
f24	**2,00e+02**	**2,00e+02**	**1,46e+01**	**2,00e+02**	**1,82e+02**	**3,07e+01**	**2,00e+02**	**2,00e+02**	**4,51e+00**
f25	2,81e+02	2,82e+02	7,31e-01	**2,81e+02**	**2,81e+02**	**2,95e-01**	2,81e+02	2,82e+02	5,49e-01
f26	1,00e+02	1,05e+02	1,83e+01	**1,00e+02**	**7,00e+01**	**4,20e+01**	1,00e+02	9,41e+01	3,14e+01
f27	3,39e+02	3,41e+02	3,62e+00	**3,37e+02**	**3,25e+02**	**1,72e+01**	3,39e+02	3,32e+02	1,39e+01

Continue page suivante

TABLEAU 2.9 : Résultats pour 100% de dim. actives - Fonctions sans rotation

	I-ABC			qABC					
	Standard			Standard			Morris		
	Médiane	Moy.	Éc.-Type	Médiane	Moy.	Éc.-Type	Médiane	Moy.	Éc.-Type
f28	8,20e+02	8,41e+02	7,98e+01	**3,24e+02**	**3,27e+02**	**1,71e+01**	3,27e+02	4,01e+02	1,74e+02

TABLEAU 2.10 : Résultats pour 50% de dim. actives - Fonctions sans rotation

	I-ABC			qABC					
	Standard			Standard			Morris		
	Médiane	Moy.	Éc.-Type	Médiane	Moy.	Éc.-Type	Médiane	Moy.	Éc.-Type
f1	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**
f2	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**
f3	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	1,79e+13	1,97e+14	5,04e+14
f4	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**
f6	3,50e-03	1,86e-02	3,12e-02	**1,00e-03**	**3,40e-03**	**5,68e-03**	1,47e-03	8,23e-03	1,37e-02
f7	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	1,21e+03	1,56e+04	7,05e+04
f8	**2,00e+01**	**2,00e+01**	**2,96e-05**	**2,00e+01**	**2,00e+01**	**4,24e-05**	**2,00e+01**	**2,00e+01**	**1,29e-05**
f9	**0,00e+00**	**0,00e+00**	**0,00e+00**	5,92e-07	2,12e-06	3,18e-06	**0,00e+00**	**9,48e-04**	**3,28e-03**
f10	**0,00e+00**	**6,91e-07**	**3,79e-06**	**0,00e+00**	**4,10e-04**	**2,25e-03**	**0,00e+00**	**6,57e-04**	**2,50e-03**
f12	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**
f13	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**
f15	1,72e-01	3,22e-01	8,58e-01	1,63e-01	1,63e-01	3,30e-02	**7,55e-02**	**9,01e-02**	**4,26e-02**
f16	1,90e-03	1,94e-03	7,23e-04	2,00e-03	2,02e-03	5,21e-04	**9,50e-04**	**1,00e-03**	**2,83e-04**
f18	**2,53e+01**	**2,53e+01**	**2,10e-03**	**2,53e+01**	**2,53e+01**	**3,32e-03**	**2,53e+01**	**2,53e+01**	**5,27e-05**
f19	2,26e-01	2,77e-01	1,79e-01	**1,87e-01**	**1,81e-01**	**4,58e-02**	2,18e-01	5,67e-01	1,77e+00
f20	2,75e-01	2,82e-01	5,45e-02	2,66e-01	2,71e-01	3,72e-02	**2,41e-01**	**2,52e-01**	**1,71e-01**
f21	**1,00e+02**	**1,20e+02**	**4,85e+01**	1,04e+02	1,17e+02	4,18e+01	1,01e+02	1,36e+02	6,72e+01
f23	2,00e+01	3,86e+01	4,02e+01	2,01e+01	2,94e+01	2,70e+01	**1,51e+01**	**2,12e+01**	**2,03e+01**
f24	**1,00e+02**	**1,00e+02**	**9,14e-03**	**1,00e+02**	**9,09e+01**	**3,03e+01**	**1,00e+02**	**8,95e+01**	**3,66e+01**
f25	**2,40e+02**	**2,35e+02**	**2,42e+01**	**2,40e+02**	**2,11e+02**	**5,10e+01**	**2,40e+02**	**1,98e+02**	**5,63e+01**
f26	**1,00e+02**	**9,45e+01**	**1,87e+01**	**1,00e+02**	**7,16e+01**	**3,90e+01**	**1,00e+02**	**8,16e+01**	**3,39e+01**
f27	**3,00e+02**	**3,02e+02**	**5,73e+00**	**3,00e+02**	**3,00e+02**	**1,85e+00**	**3,00e+02**	**3,01e+02**	**3,54e+00**
f28	**3,00e+02**	**2,86e+02**	**5,07e+01**	**3,00e+02**	**2,83e+02**	**5,20e+01**	**3,00e+02**	**2,63e+02**	**7,64e+01**

TABLEAU 2.11 : Résultats pour 25% de dim. actives - Fonctions sans rotation

	I-ABC			qABC					
	Standard			Standard			Morris		
	Médiane	Moy.	Éc.-Type	Médiane	Moy.	Éc.-Type	Médiane	Moy.	Éc.-Type
f1	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**
f2	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**
f3	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	1,99e+10	2,81e+10	3,20e+10
f4	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**
f6	**8,56e-04**	**1,54e-03**	**1,93e-03**	3,15e-03	5,38e-03	8,31e-03	2,10e-03	4,55e-03	7,14e-03
f7	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	1,51e+02	3,57e+02	8,99e+02
f8	**2,00e+01**	**2,00e+01**	**5,73e-05**	**2,00e+01**	**2,00e+01**	**6,68e-05**	**2,00e+01**	**2,00e+01**	**4,51e-06**
f9	**0,00e+00**	**0,00e+00**	**0,00e+00**	2,42e-08	1,35e-06	3,86e-06	**0,00e+00**	**1,94e-05**	**1,03e-04**
f10	**0,00e+00**	**3,28e-04**	**1,38e-03**	**0,00e+00**	**1,23e-03**	**3,99e-03**	**0,00e+00**	**9,86e-04**	**2,56e-03**
f12	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**
f13	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**
f15	5,21e-02	5,60e-02	4,79e-02	1,04e-01	1,17e-01	4,62e-02	**0,00e+00**	**8,67e-03**	**2,40e-02**
f16	3,17e-03	3,21e-03	1,96e-03	7,66e-03	7,83e-03	1,80e-03	**7,44e-04**	**7,41e-04**	**3,78e-04**
f18	**1,22e+01**	**1,06e+01**	**4,19e+00**	**1,22e+01**	**1,07e+01**	**3,34e+00**	**1,22e+01**	**8,09e+00**	**5,49e+00**
f19	**2,91e-02**	**5,87e-02**	**8,75e-02**	4,24e-02	4,84e-02	2,59e-02	4,01e-02	9,20e-02	2,41e-01
f20	7,84e-02	7,72e-02	3,30e-02	8,07e-02	7,80e-02	3,99e-02	**4,80e-05**	**3,45e-02**	**4,74e-02**
f21	**1,00e+02**	**1,34e+02**	**5,38e+01**	1,15e+02	1,25e+02	3,24e+01	1,20e+02	1,42e+02	4,31e+01
f23	6,11e-02	1,94e+00	2,25e+00	6,53e+00	6,88e+00	3,87e+00	**3,09e-04**	**3,56e-02**	**7,26e-02**

Continue page suivante

TABLEAU 2.11 : Résultats pour 25% de dim. actives - Fonctions sans rotation

	I-ABC			qABC					
	Standard			Standard			Morris		
	Médiane	Moy.	Éc.-Type	Médiane	Moy.	Éc.-Type	Médiane	Moy.	Éc.-Type
f24	1,00e+02	8,42e+01	4,09e+01	1,00e+02	8,22e+01	3,31e+01	**8,39e+00**	**4,24e+01**	**4,62e+01**
f25	2,00e+02	1,77e+02	3,96e+01	2,00e+02	1,59e+02	4,58e+01	**1,30e+02**	**1,51e+02**	**4,75e+01**
f26	8,59e+01	6,24e+01	4,24e+01	1,52e+01	3,47e+01	3,96e+01	**1,24e+00**	**1,76e+01**	**3,42e+01**
f27	**3,00e+02**	**2,89e+02**	**4,49e+01**	**3,00e+02**	**2,84e+02**	**5,05e+01**	**3,00e+02**	**2,82e+02**	**5,51e+01**
f28	**1,00e+02**	**4,08e+02**	**3,58e+02**	1,22e+02	1,98e+02	1,60e+02	1,50e+02	2,52e+02	2,38e+02

Bien que les optima des fonctions soient plus aisément atteignables lorsqu'aucune rotation n'est appliquée, nous pouvons remarquer que l'ajout de la méthode Morris à la métaheuristique qABC fournit des résultats équivalents, lorsque toutes les dimensions sont actives, et améliore significativement ses performances initiales, lorsque le nombre de dimensions actives diminue et que l'optimum n'est pas atteint. À 100% de dimensions actives (Tableau 2.9), qAbc trouve la meilleure erreur médiane pour toutes les fonctions. Néanmoins, la version « Morris » obtient des résultats identiques sur huit fonctions. Mais dès lors que le nombre de dimensions actives diminue (Tableaux 2.10 et 2.11), qABC-Morris obtient la plus faible erreur médiane pour la majorité des fonctions, améliorant l'algorithme d'origine mais donnant aussi de meilleurs résultats que I-ABC.

De plus, le nombre d'évaluations défini par le *benchmark* ne permet pas de discriminer suffisamment les deux versions d'un même algorithme. Il est en effet intéressant de suivre l'évolution de la plus faible erreur trouvée au cours des différentes évaluations de l'algorithme, afin de noter si le fait de prendre en compte l'influence d'une dimension dans le processus de recherche accélère la convergence.

Les figures 9 à 12, présentées en annexe B, illustrent la vitesse de convergence des deux versions de l'algorithme qABC, respectivement appliquées pour 50% et 25% de dimensions explicites. Elles représentent l'erreur médiane pour chacun des algorithmes à intervalles réguliers d'évaluations.

Pour 50% de dimensions actives, il ne se dégage pas de comportement général pour l'ensemble des fonctions étudiées. En effet, à l'exception des fonctions $f3$ et $f7$ pour lesquelles nous pouvons constater aucune convergence de l'algorithme qABCMorris, l'allure générale de la convergence est identique pour les deux algorithmes. Nous pouvons néanmoins remarquer que, lorsque les deux algorithmes trouvent l'optimum, la version qABCMorris converge plus rapidement.

Nous constatons également qu'à 25% de dimensions actives, qABCMorris converge plus rapidement et, en particulier sur certaines fonctions telles que $f14$, $f20$, $f22$, $f23$ ou $f24$, cette convergence s'étend au long du déroulement de l'algorithme. Ce dernier continue à converger alors que la version d'origine stagne sur un optimum local.

Les résultats confirment nos attentes. En effet, l'intérêt d'utiliser la méthode de Morris est d'améliorer la convergence de l'algorithme, notamment lorsque la fonction possède plusieurs dimensions à faible influence vis-à-vis des autres.

2.4 Conclusion

Dans ce chapitre, nous avons présenté une amélioration générique pour l'algorithme de colonie d'abeilles artificielles, ainsi que pour plusieurs de ses dérivés, à travers l'intégration d'une méthode d'analyse de sensibilité, la méthode de Morris. Cette méthode permet d'explorer plus judicieusement l'espace de recherche, en privilégiant les dimensions qui ont une influence significative sur la fonction à optimiser. Connaissant l'influence de chaque dimension sur la fonction objectif, la métaheuristique évitera de réaliser des évaluations de la fonction objectif, peu pertinentes en termes d'efficacité de recherche. En effet, cela réduit les chances de se perdre en parcours de dimensions dont l'impact est très faible. Les résultats ont montré que la version « Morris » des algorithmes surpasse l'originale en termes d'erreur et de rapidité de convergence, lorsque le nombre de ces dimensions non influentes est élevé par rapport à la dimension totale du problème, et particulièrement lorsqu'aucune rotation n'est appliquée.

Un intérêt évident dans l'utilisation d'une telle méthode est qu'elle ne nécessite pas d'évaluation supplémentaire de la fonction objectif (ce qui est généralement le plus coûteux). En effet, elle utilise les calculs effectués au cours de son déroulement pour évaluer le poids de chaque dimension.

De plus, de par sa simplicité et son comportement similaire à celui de l'algorithme ABC, nous avons montré que la méthode de Morris peut s'intégrer dans plusieurs variantes de l'algorithme qui respectent la contrainte donnée par l'équation (1.41). Si la version de l'algorithme est compatible avec l'équation des effets élémentaires (1.57) de la méthode de Morris (i.e. le décalage peut s'exprimer selon $X'^{i}_{j} = X^{i}_{j} + \Delta$), comme pour [Kiran et Findik, 2015; Kiran et collab., 2015], par exemple, il est alors fort probable que l'ajout de la méthode de Morris dans cet algorithme améliore ses performances. La performance globale de l'algorithme ainsi créé dépend essentiellement de la performance globale de la version originale de l'algorithme étudié.

Pour aller plus loin dans l'intérêt d'intégrer une méthode d'analyse de sensibilité au sein d'une métaheuristique, il est nécessaire d'étendre le principe à d'autres stratégies de recherche. Le modèle présenté ne convient pas aux métaheuristiques modifiant plusieurs dimensions simultanément. Or, plusieurs variantes de l'algorithme ABC, comme celles utilisant un taux de modification [Akay et Karaboga, 2012; Ozkis et Babalik, 2013], une méthode de recherche mémétique [Fister et collab., 2012], des méthodes hybrides (opérateurs génétiques [Kiran et Gündüz, 2012; Kumar et collab., 2013] ou un décalage inspiré de l'algorithme PSO [Imanian et collab., 2014]) améliorent la version originale en réalisant une recherche multi-dimensionnelle. Alors, l'impact du décalage ne peut plus s'évaluer par la méthode des effets élémentaires.

Dans l'objectif d'appliquer une méthode d'analyse de sensibilité dans de tels cas et pour d'autres typologies de métaheuristiques, il est nécessaire d'étudier un autre type de méthode d'analyse de

sensibilité, pouvant évaluer le poids de l'ensemble des dimensions suite à une série de décalages multi-dimensionnels.

Chapitre 3

Élaboration de la méthode d'analyse de sensibilité NN-LCC pour les métaheuristiques à déplacement multidimensionnel

Ce chapitre est une généralisation du chapitre précédent, à savoir l'intégration d'une méthode d'analyse de sensibilité dans des métaheuristiques à déplacement multidimensionnel. Pour cela, nous présentons la création d'une nouvelle méthode d'analyse de sensibilité, la méthode NN-LCC, et l'évaluation de la pertinence de son intégration dans deux métaheuristiques différentes.

Notions abordées dans ce chapitre :

- *Coefficients de corrélation linéaire ;*
- *Colonie d'abeilles artificielles ;*
- *Évolution différentielle.*

3.1 Introduction

Dans le chapitre précédent, nous avons présenté comment une méthode d'analyse de sensibilité, la méthode de Morris, couplée à une métaheuristique pour l'optimisation continue, l'algorithme de colonie d'abeilles artificielles (ABC), permet d'améliorer les résultats de recherche de l'optimum. En focalisant principalement sa recherche sur les dimensions qui possèdent une influence significative sur la fonction objectif, la métaheuristique converge plus rapidement. Les calculs de la moyenne absolue et de l'écart-type des effets élémentaires de la méthode de Morris permettent de déterminer l'influence de chaque paramètre. La méthode de Morris s'intègre particulièrement bien dans l'algorithme ABC car ce dernier génère aléatoirement une nouvelle solution en décalant une dimension. Cela correspond au processus itératif de calcul des effets élémentaires de la méthode de Morris. Chaque évaluation réalisée par l'algorithme, améliorante ou non, est injectée dans le calcul des effets élémentaires et permet d'affiner l'influence des dimensions sur la fonction objectif.

Cependant, parmi l'ensemble des métaheuristiques présentées en 1.1, la génération d'un nouvel individu par décalage unidimensionnel est un cas particulier. Certaines versions de l'algorithme ABC [Akay et Karaboga, 2012; Li et collab., 2012] possèdent un algorithme de recherche par décalage multi-dimensionnel. D'autres, comme la recherche tabou [Chelouah et Siarry, 2000b], l'évolution différentielle [Price et collab., 2005] ou certains algorithmes de la famille d'intelligence en essaim (ACO [Dréo et Siarry, 2007], PSO [El Dor et collab., 2012; Kennedy et Eberhart, 1995]) recherchent un voisin dans une hyper-sphère, se déplaçant selon plusieurs dimensions à la fois.

Aussi, parmi l'ensemble des méthodes d'analyse de sensibilité abordées en 1.4, la méthode des indices de Sobol est la méthode la plus précise. Cette méthode gère les fonctions complexes mais nécessite un nombre important d'évaluations pour l'estimation des indices par la méthode de Monte-Carlo. Bien que les métamodèles puissent être utilisés afin de réduire ces évaluations, cette méthode reste encore trop coûteuse pour être intégrée dans un algorithme d'optimisation. Certains types de métamodèles conduisent à une expression analytique des indices de sensibilité, en évitant l'estimation coûteuse des indices par la méthode de Monte-Carlo (Chaos polynomial [Sudret, 2008], modèles linéaires complexes [Jourdan, 2012]). Néanmoins, ces méthodes étant basées sur un système probabiliste ou/et sur un plan d'expérience, elles ne peuvent s'inscrire dans notre contexte, où les évaluations sont données par le comportement de la métaheuristique. La méthode de Morris gère aussi les fonctions complexes et nécessite peu d'évaluations. Comme vu précédemment, elle possède en revanche la contrainte de déplacement unidimensionnel. Possédant le même ordre de coût calculatoire que la méthode de Morris, les méthodes de coefficients de corrélation linéaire et de coefficients de corrélation linéaire basés sur les rangs (vus en 1.5) pourraient convenir. Elles présentent néanmoins une contrainte forte sur le modèle, qui doit être linéaire ou monotone.

Par conséquent, afin de généraliser la première approche d'intégration d'une méthode d'analyse de sensibilité au sein d'une métaheuristique et de pouvoir l'appliquer à un ensemble plus important de métaheuristiques, il est nécessaire de construire une nouvelle méthode d'analyse de sensibilité. Cette dernière devra pouvoir détecter l'influence de chacune des dimensions décalées, en respectant les contraintes suivantes :

— la méthode doit s'appliquer à des fonctions complexes, dans le cadre de l'optimisation difficile ;
— la méthode ne doit pas être régie par un plan d'expérience, l'ensemble des points étant donné par l'exécution de la métaheuristique ;
— la méthode doit fournir des résultats significatifs sans nécessiter un nombre trop élevé de points et de leurs évaluations.

Pour que cette analyse de sensibilité soit profitable à la métaheuristique dans laquelle elle sera intégrée, elle doit pouvoir déterminer les dimensions peu ou pas influentes et les classer entre elles.

Dans notre contexte, et respectant ces contraintes, nous proposons une méthode d'analyse de sensibilité, NN-LCC [Loubière et collab., 2016a,b], qui combine une méthode de régression (linéaire ou sur les rangs) appliquée sur des voisinages restreints. Nous appliquons sur l'ensemble de ces voisinages une analyse du même type que celle des effets élémentaires pour la méthode de Morris, afin d'obtenir une analyse de sensibilité globale d'un modèle supposé non linéaire. Cette analyse fournit le poids de chaque variable sur le modèle.

Nous présentons la méthode d'analyse de sensibilité développée ainsi que plusieurs tests sur la cohérence de ses résultats, pour un ensemble de fonctions usuelles du domaine. Puis nous détaillons la méthodologie afin de l'intégrer dans une métaheuristique, et enfin les performances des algorithmes ainsi obtenus sur le jeu de test de la conférence CEC 2013.

3.2 Méthodes d'analyse de sensibilité des plus proches voisins

L'objectif ici est de s'inspirer d'une méthode linéaire globale d'analyse de sensibilité pour construire une méthode applicable dans un contexte non linéaire. Cette dernière possédera les propriétés de faible coût en nombre d'évaluations, ainsi que de simplicité de mise en œuvre, en vue de son inclusion dans une métaheuristique.

3.2.1 Description de l'algorithme

Nous allons nous inspirer des méthodes de corrélation linéaire et de corrélation sur les rangs abordées en 1.5. Nous retenons ces deux méthodes afin de vérifier si une méthode adaptée au contexte monotone donne de meilleurs résultats qu'une méthode dédiée au contexte affine. Afin de transposer une méthode globale d'analyse de sensibilité d'un contexte linéaire vers un contexte non linéaire, il

est intéressant d'évaluer localement les coefficients de corrélation. Partant d'un ensemble de points et de leur évaluation par la fonction objectif, plusieurs voisinages locaux sont définis. Selon la méthode choisie, les coefficients de corrélation linéaire (NN-LCC) ou bien les coefficients de corrélation sur les rangs (NN-RCC) sont calculés, par voisinage de points.

Considérant un ensemble de N points déjà connus, appartenant à un espace de recherche de dimension D, on choisit aléatoirement k points ($k < N$), définissant les centres des voisinages. Un voisinage est constitué des p plus proches voisins de son centre.

Pour chaque voisinage de points, l'équation (1.55) évalue la corrélation linéaire de chaque dimension. Ainsi, pour le i^{ieme} voisinage et la j^{ieme} dimension, le coefficient de corrélation linéaire correspondant est défini selon l'équation (3.1) :

$$LCC_j^i = \rho(X_j, Y), i \in \{1, ..., k\}, j \in \{1, ..., D\}. \tag{3.1}$$

où X désigne un point du i^{ieme} voisinage et Y, son image par la fonction objectif. Une matrice de coefficients est alors construite. De la même manière, la matrice des coefficients de corrélation sur les rangs est construite selon l'équation (3.2) :

$$RCC_j^i = \rho_j^s = \rho^s(X_j, Y) = \rho(R_{X_j}, R_Y), i \in \{1, ..., k\}, j \in \{1, ..., D\}. \tag{3.2}$$

La moyenne absolue (m_j^*) et l'écart-type (s_j) sont ensuite calculés par dimension (i.e. par colonne de la matrice), selon la méthode de Morris. L'influence de la j^{ieme} dimension est alors donnée par la distance pondérée d_j du couple (m_j^*, s_j) à l'origine (3.3) :

$$d_j = \sqrt{m_j^{*2} + \delta s_j^2}\ ,\ \delta > 0. \tag{3.3}$$

Le paramètre δ permet de privilégier le caractère non linéaire d'une dimension, par rapport à son caractère linéaire, en augmentant son influence. L'objectif est de favoriser le comportement non linéaire d'une dimension. Son poids sera plus élevé et cela favorisera sa sélection. Une étude de l'influence de ce paramètre reste à faire néanmoins, il est fortement dépendant du problème à résoudre. En effet, même si Campolongo *et al.* ont montré dans [Campolongo et collab., 2007] que m_j^* est une bonne approximation des indices de Sobol totaux d'une variable d'entrée, nous incluons l'écart-type (plus ou moins, en fonction de δ). Une valeur forte pour s_j indiquerait une fonction multimodale (Figure 3.1).

Le poids de la j^{ieme} dimension (w_j) est enfin donné par :

$$w_j = d_j / (\sum_{k=1}^{D} d_k). \tag{3.4}$$

La procédure complète est illustrée par l'algorithme 3.1.

Algorithme 3.1 : Algorithme NN-LCC.

1 : Initialiser N points aléatoires et leurs évaluations
2 : Choisir aléatoirement $k < N$ points pour définir les voisinages
3 : **pour** $i \leftarrow 1$ *à* k **faire**
4 : | V $\leftarrow p$ plus proches voisins de X^i
5 : | **pour** $j \leftarrow 1$ *à* D **faire**
6 : | | $LCC_j^i = |\rho(X_j, Y)|,\ X \in V$
7 : | **fin**
8 : **fin**
9 : **pour** $j \leftarrow 1$ *à* D **faire**
10 : | Calculer $\mu_j^*,\ \sigma_j,\ d_j$, selon (1.60), (1.59) et (3.3)
11 : **fin**
12 : **pour** $j \leftarrow 1$ *à* D **faire**
13 : | Calculer w_j, selon (3.4)
14 : **fin**

La recherche des p plus proches voisins (ligne 4) peut nécessiter un certain temps de calcul. Néanmoins, considérant le fait qu'un appel à la fonction objectif est l'opération la plus coûteuse, cet algorithme ne requiert pas d'évaluation supplémentaire de la fonction objectif. De la même manière que l'algorithme ABC-Morris, il utilise celles déjà effectuées par la métaheuristique.

3.2.2 Illustration

La figure 3.1 illustre le principe de notre méthode. Considérons une fonction admettant une solution X de dimension 3. La partie gauche de la figure illustre le comportement de chaque dimension $X_i,\ i = \{1, 2, 3\}$, sur la fonction. Les points rouges représentent les voisinages et les flèches représentent les coefficients de corrélation. Le graphique de droite représente, dans le référentiel (moyenne absolue, écart-type), la corrélation calculée pour chaque dimension.

La première dimension X_1 n'a pas (ou très peu) d'influence sur le résultat de la fonction. Pour l'ensemble des voisinages, la moyenne et l'écart-type des coefficients de corrélation ont de faibles valeurs. La dimension X_2 est plus influente, elle possède une forte moyenne absolue et un faible écart-type. Cela signifie que l'influence de cette variable est linéaire. La dimension X_3 possède une faible valeur de moyenne absolue et une forte valeur d'écart-type. Cela signifie que l'influence de cette variable est fortement non linéaire.

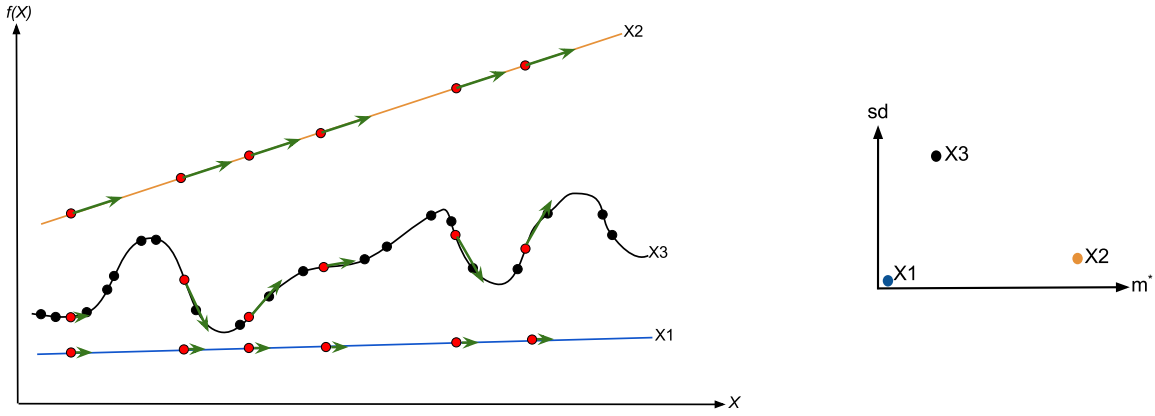

FIGURE 3.1 : Projection de chaque dimension et leur poids correspondant.

Les poids de chaque dimension seront représentés par la distance de chaque point (moyenne absolue, écart-type) à l'origine, divisée par la somme des distances.

3.2.3 Tests et discussion

Dans un premier temps, nous allons éprouver la méthode sur plusieurs fonctions de tests classiques de l'analyse de sensibilité. Le but est de vérifier son efficacité dans la détection des poids des variables pour ces fonctions, en comparant les résultats avec les poids connus (donnés par les indices totaux de Sobol) et leurs estimations par la méthode de Morris.

3.2.4 Test de la méthode d'analyse de sensibilité

Nous allons tester notre méthode d'analyse de sensibilité par corrélation linéaire des plus proches voisins (NN-LCC) sur quatre fonctions de test (voir Tableau 3.1) de l'analyse de sensibilité [Surjanovic et Bingham].

3.2.4.1 Présentation

Les poids théoriques p_j (cf équation (1.63)) des variables X_j sont connus pour ces fonctions. Ils seront comparés avec les poids estimés w_j (cf équation (3.4)) en utilisant la mesure de score donnée par l'équation (3.5), qui définit une distance euclidienne (au carré) du poids trouvé à la valeur connue :

$$score = \frac{1}{D}\sum_{j=1}^{D}(p_j - w_j)^2. \tag{3.5}$$

L'objectif est double : évaluer de manière correcte les poids (ou au moins leurs rangs) des variables d'entrée influentes pour ces fonctions, mais aussi comparer les résultats avec des méthodes existantes (méthode de Morris et indices de Sobol).

TABLEAU 3.1 : Fonctions de test.

Fonction	Formulation	Domaine	Dim.	Poids théoriques %
Polynomial	$f(X) = X_1 * X_2$	$[0,1]$	2	$50, 50$
Welch	$f(X) = \frac{5X_2}{1+X_1}$	$[-0.9,1]$	2	$37, 63$
Ishigami	$f(X) = \sin(X_1) + 7\sin^2(X_2) + 0.1\ \sin(X_1)\sin^4(X_3)$	$[-\pi,\pi]$	3	$44.8, 35.6, 19.6$
Sobol g-function	$f(X) = \prod_{i=1}^{D} \frac{\lvert 4X_i - 2\rvert + c_i}{1+c_i}$			
v1	$c = (0.01, 0.01, 100, 100)$	$[0,1]$	4	$49.993, 49.993, 0.007, 0.007$
v2	$c = (0.01, 1, 100, 100)$		4	$73.72, 26.26, 0.01, 0.01$
v3	$c = (0.1, 0.1, 0.5, 100, 100, 100, 100, 100, 100, 100)$		10	$38.657, 38.657, 22.648, 0.005, \ldots$

3.2.4.2 Protocole de test

Les paramètres de la méthode sont k et p, respectivement, le nombre de voisinages et le nombre de plus proches voisins composant le voisinage d'un point.

Afin d'étudier l'impact du choix de ces paramètres, nous avons utilisé un plan d'expérience composé de cinquante ensembles de paramètres, selon les variations suivantes : considérant N points, $N \in [200, 1000]$, nous définissons k comme un pourcentage du nombre de points, variant de 10% à 50% de N. Le paramètre p dépend de la dimension (approximativement égal à $10D$, cf Figure 1.12). Enfin, nous faisons varier les dimensions des fonctions de test de 2 à 10 (cf Tableau 3.1).

Le plan d'expérience est construit selon un *space-filling design* [Bartz-Beielstein et Preuss, 2007; Jourdan et Franco, 2010]. Chaque ensemble de paramètres est appliqué aux fonctions définies dans le tableau 3.1. Les N points sont choisis aléatoirement dans les domaines de définition de chaque fonction.

La figure 3.2 illustre l'impact de chaque paramètre de la méthode NN-LCC sur le score calculé.

Nous pouvons remarquer que, pour les fonctions Ishigami et Sobol, un nombre de voisins supérieur à 60 donne de meilleurs scores ; néanmoins, nous observons une stagnation au-dessus de 80. Pour les autres fonctions, le nombre de voisins ne semble pas avoir d'influence particulière sur les scores calculés.

Pour une comparaison équitable avec la méthode de Morris, le protocole de test est construit de sorte que le nombre d'évaluations soit identique à celui des méthodes NN-LCC et NN-RCC. On effectue dix décalages le long de chaque dimension et le nombre initial de points aléatoirement choisis est $N/(10 * D)$.

3.2.4.3 Résultats

La figure 3.3 montre que les méthodes initiales, de coefficients de corrélation linéaire et basés sur les rangs (LCC et RCC) fournissent de mauvaises estimations des poids. La méthode de Morris donne globalement le meilleur score. Quant aux méthodes NN-LCC and NN-RCC, elles donnent des résultats comparables. La méthode NN-RCC nécessitant plus de calculs que la méthode NN-LCC (calculs des rangs), nous retiendrons alors seulement la méthode NN-LCC pour le détail des évaluations des poids

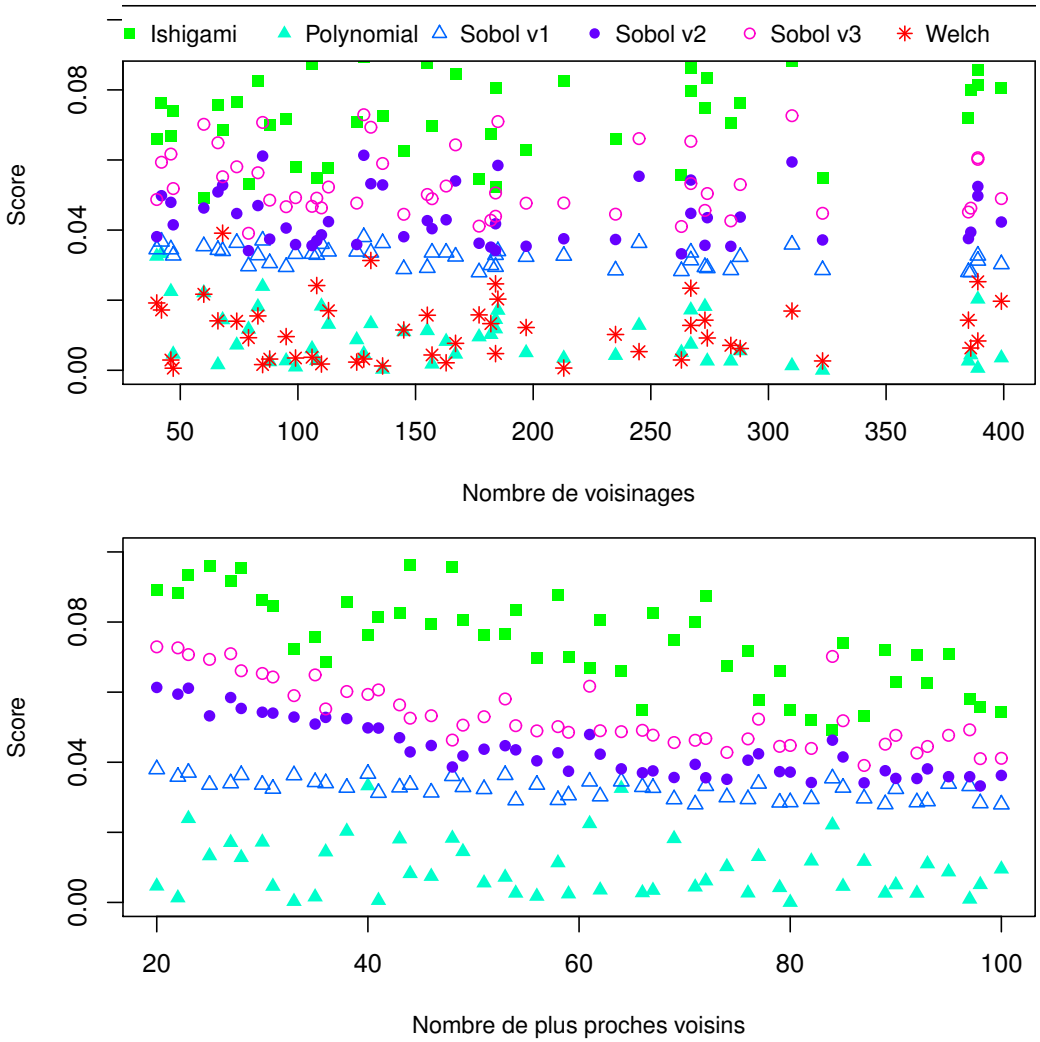

FIGURE 3.2 : Influence de chaque paramètre (k, p) de la méthode NN-LCC sur le score.

et les comparaisons futures (Figures 3.4, 13 et 14). Le score obtenu par la méthode de Morris est meilleur que les scores des méthodes NN-LCC et NN-RCC, même si leurs moyennes (illustrées par les croix) présentent peu de différence. La p-valeur du test de Student entre les méthodes de Morris et NN-LCC est de 3.27%, nous ne pouvons pas conclure sur une différence significative (nous constatons, sur la figure 3.3, que la distribution des scores est suffisamment régulière pour utiliser le test de Student).

Le score obtenu par la méthode NN-LCC semble globalement moins bon ; néanmoins, comme nous le détaillons ci-après (Figure 3.4), ce résultat diffère selon les fonctions.

La figure 3.4 détaille les résultats des évaluations des poids de toutes les variables, pour chaque fonction.

Selon les figures 13 et 14, en annexe C, lorsque toutes les variables d'entrée (i.e. dimensions) ont un impact significatif sur le résultat de la fonction (fonctions Ishigami, Polynomial et Welch), la méthode NN-LCC fournit une meilleure évaluation des poids que la méthode de Morris. Pour la

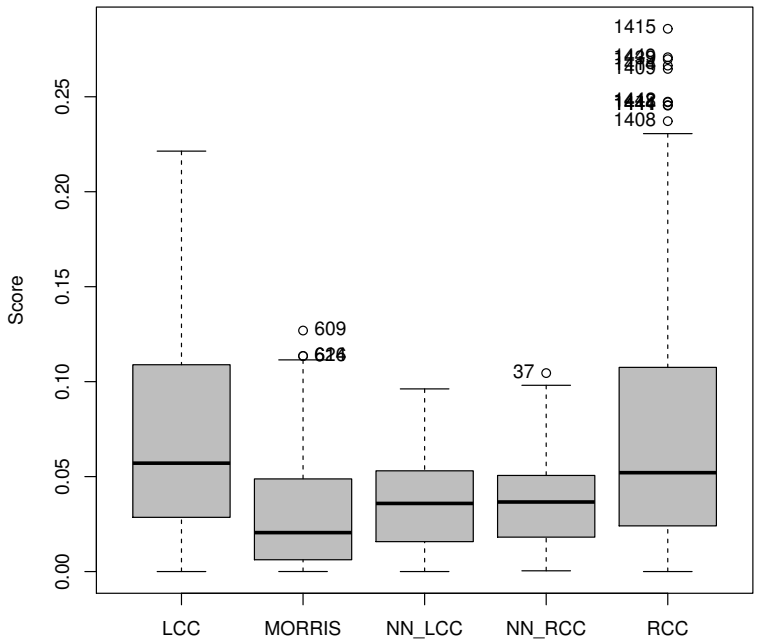

FIGURE 3.3 : Scores globaux de chaque méthode.

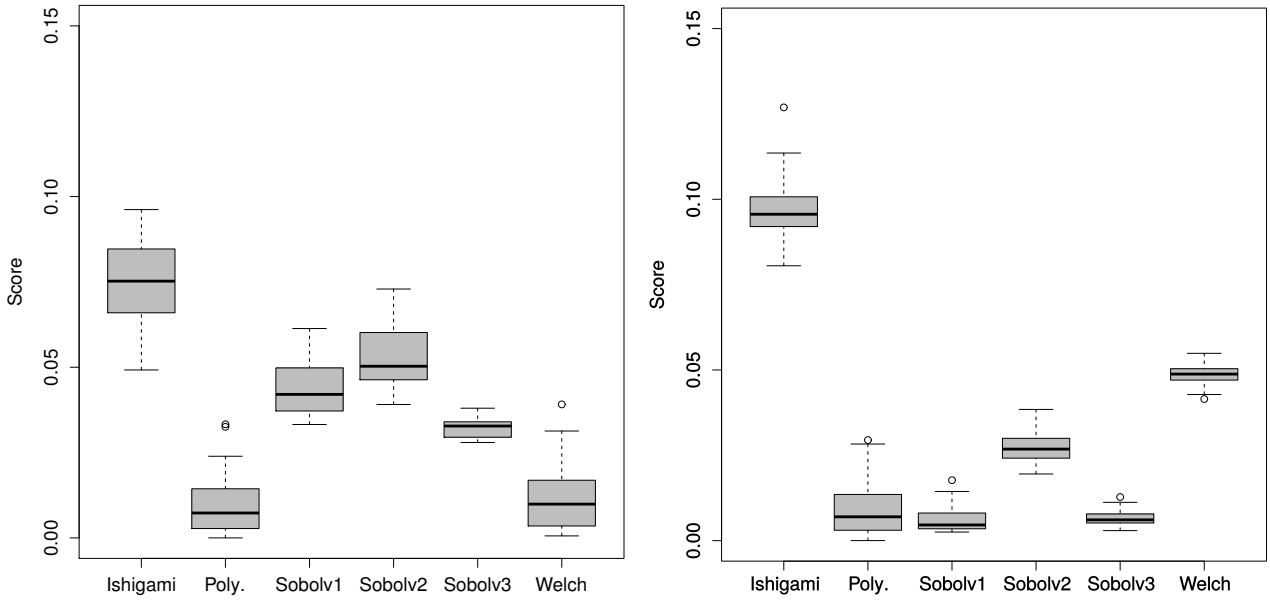

FIGURE 3.4 : Scores des méthodes NN-LCC (à gauche) et Morris (à droite) pour chaque fonction.

fonction Polynomial, la plus simple, les deux méthodes fournissent des résultats comparables. Pour la fonction Ishigami, la fonction la plus complexe, la méthode NN-LCC évalue mieux les poids des variables X_1 et X_2, mais tend à légèrement surestimer celui de X_3, en comparaison de la méthode de

Morris. Pour la fonction Welch, l'influence est correctement évaluée par la méthode NN-LCC.

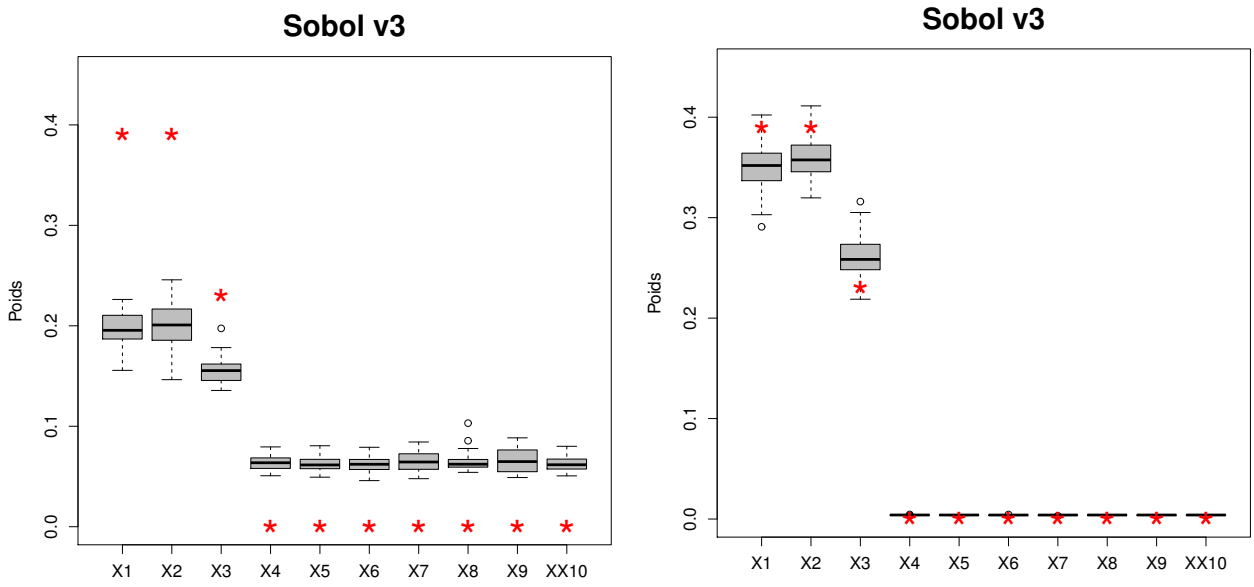

FIGURE 3.5 : Scores des méthodes NN-LCC (à gauche) et Morris (à droite) pour Sobol v3.

Cependant, lorsqu'il existe des variables dont le poids est négligeable pour une fonction (Sobol), NN-LCC surestime les poids des dimensions possédant une influence quasi nulle et sous-estime les plus influentes (excepté pour la variable X_2 de la fonction de Sobol). Dans ces cas, la méthode de Morris évalue plus précisément l'influence des variables d'entrée. Ce comportement est particulièrement mis en évidence avec la fonction de Sobol v3 (Figure 3.5).

La méthode de Morris effectue une série de décalages unidirectionnels, dimension par dimension. Ainsi, lorsqu'une variable non (ou peu) influente est décalée, la méthode identifie plus précisément une faible variation du résultat de la fonction. Dans le cas d'un décalage multidirectionnel (des méthodes NN-LCC et NN-RCC), l'influence de chaque dimension sur le résultat ne peut être détectée aussi distinctement.

Plus le nombre de dimensions d'influence négligeable augmente, moins l'évaluation de l'influence des variables d'entrée par la méthode NN-LCC est précise. Afin d'illustrer ce comportement, nous avons augmenté artificiellement le nombre de dimensions, en utilisant un certain nombre de variables « muettes » (de poids théorique nul).

Par exemple, la fonction Welch est définie à deux dimensions. Le poids théorique de la variable d'entrée X_2 pour cette fonction est 0.63. La figure 3.6 illustre l'évolution du poids calculé pour la variable X_2 lorsque la dimension du problème varie de 5 à 30, les dimensions supérieures à 2 possédant une influence nulle. Lorsque le nombre de dimensions dépasse 25, le poids de X_2 tombe en dessous de $0,2$. Ce comportement, comme observé pour la fonction Sobol v3, est la conséquence de la répartition

de l'effet sur toutes les dimensions.

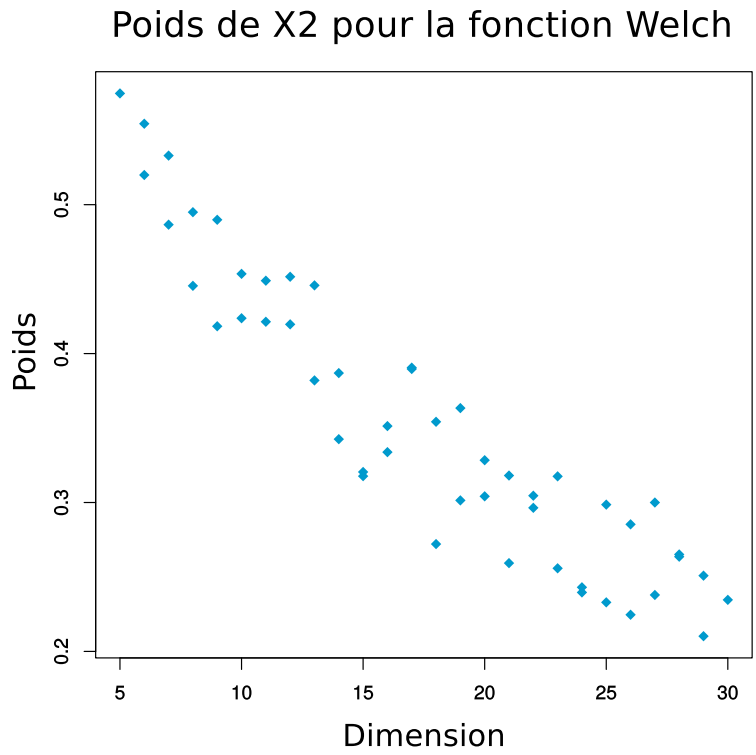

FIGURE 3.6 : Évolution du poids de la variable X_2 pour la fonction Welch selon le nombre de dimensions.

Malgré ce comportement, les rangs des poids des variables d'entrée évalués par la méthode sont respectés. La méthode permet de classer correctement les dimensions entre elles, de la même manière que la méthode NN-RCC, plus coûteuse en calculs. Elle permettra donc, en vue de son intégration dans une métaheuristique, de prioriser la recherche le long des dimensions les plus prometteuses.

3.3 Intégration de la méthode aux métaheuristiques

Nous proposons ici d'inclure la méthode d'analyse de sensibilité NN-LCC dans des métaheuristiques modifiant plusieurs dimensions simultanément.

L'idée est, dans un premier temps, de se servir d'une partie des individus générés par la métaheuristique et de leur évaluation par la fonction objectif, afin de déterminer les poids de chaque dimension. Pour effectuer le calcul de ces poids, la méthode nécessite N points. Nous laissons alors la métaheuristique s'exécuter en gardant les N premiers points et leurs évaluations. Ainsi, l'algorithme 3.1 calcule le vecteur des poids de chaque dimension. Ces informations sont ensuite conservées pour l'étape suivante.

Dans un second temps, il s'agit de modifier l'équation de génération de la métaheuristique considérée. La probabilité de sélection d'une dimension est généralement un paramètre de l'algorithme, identique pour toutes les dimensions. Nous proposons de le remplacer par le vecteur des poids calculés précédemment, afin de favoriser la sélection des dimensions les plus influentes. Par la suite, l'intégration des poids des dimensions dans la génération d'un nouvel individu est détaillée dans deux exemples : une version modifiée de l'algorithme de colonies d'abeilles artificielles, ABCMR, et l'algorithme d'évolution différentielle (DE).

L'objectif est d'observer si l'ajout de la méthode fournit de meilleurs résultats que l'algorithme de base, et de comparer cette méthode avec les résultats donnés par l'algorithme ABC-Morris.

3.3.1 Intégration de la méthode dans l'algorithme ABCMR

Présenté à la section 1.3.3 du chapitre 1, l'algorithme ABC avec taux de modification [Akay et Karaboga, 2012] apporte plusieurs améliorations à l'algorithme ABC : un coefficient de décalage et un taux de modification. Ce taux de modification (*modification rate*, MR) introduit dans l'algorithme un décalage de plusieurs dimensions de la solution, selon un tirage aléatoire, lors de la génération d'un nouvel individu.

Basée sur l'équation (1.50), l'application d'une méthode d'analyse de sensibilité va faire évoluer la sélection aléatoire de dimensions à modifier. De la même manière que pour l'algorithme ABC-Morris, l'idée est de favoriser la sélection des dimensions influentes (selon le poids calculé par la méthode d'analyse de sensibilité). L'équation de génération d'un nouvel individu (1.50) est alors transformée en (3.6) :

$$V_j^i = \begin{cases} X_j^i + \phi_{ij}(X_j^i - X_{kj}) & , \text{ si } rand(0,1)/D < w_j, \\ X_j^i & , \text{ sinon.} \end{cases} \tag{3.6}$$

Le nombre aléatoire généré pour une dimension, que l'on va comparer au poids de cette dimension, est égal à $rand(0,1)/D$. Il a été défini, selon le postulat que, si pour une solution X^i, toutes les dimensions X_j^i ont le même poids, ce dernier sera alors de $1/D$. Cela permet de prendre en compte le phénomène de répartition de l'influence d'une dimension sur des dimensions de moindre poids, comme expliqué précédemment.

3.3.2 Intégration de la méthode dans l'algorithme DE

Nous avons vu, dans le chapitre 1, section 1.2.2, que l'algorithme proposé par Storn et Price [Storn et Price, 1997] engendre de nouveaux individus en utilisant les opérations de mutation (1.11) et de croisement (1.2). Comme la métaheuristique précédente, l'algorithme DE effectue un déplacement

multi-dimensionnel. Le choix des dimensions à décaler se fait de manière « aveugle », dans le sens où il dépend du paramètre de taux de croisement (CR). De la même manière que précédemment, l'intégration de l'utilisation des poids des dimensions dans la méthode de croisement binaire se fait selon l'équation (3.7) :

$$u^i_{j,G+1} = \begin{cases} v^i_{j,G+1} & , \text{ si } rand(0,1)/D \leq w_j \text{ ou } j = rand_j, \\ X^i_{j,G} & , \text{ sinon.} \end{cases} \tag{3.7}$$

3.4 Tests et résultats

Nous allons présenter, dans cette section, les résultats de l'application de la méthode d'analyse de sensibilité par coefficients de corrélation linéaire modifiée (NN-LCC). Nous avons intégré la méthode dans les algorithmes ABCMR et DE testés sur les fonctions du *benchmark* CEC 2013 [Li et collab., 2013]. L'objectif ici est d'étudier l'efficacité de l'ajout de la méthode dans une métaheuristique qui modifie un sous-ensemble de dimensions. Il s'agit en premier lieu, lorsque l'ensemble des dimensions sont actives, de constater que l'ajout de la méthode ne dégrade pas les résultats de la métaheuristique. Puis, dans un second temps, nous vérifierons que l'ajout de la méthode est profitable lorsqu'un ensemble de dimensions n'a pas d'influence sur la fonction objectif.

3.4.1 Paramétrage

L'algorithme NN-LCC a été intégré dans l'algorithme ABCMR [Akay et Karaboga, 2012]. Ce dernier a été paramétré avec les valeurs suivantes : $SN = 25$ et $MR = 0,4$.

Pour son intégration dans l'algorithme DE, nous avons pris comme schéma le triplet DE/rand/1 (1.1). Les paramètres de l'algorithme sont $NP = 25$, $F = 0,5$ et $CR = 0,9$ [S. Das, 2016].

Les tests sont réalisés sur le *benchmark* CEC 2013, en dimensions $D = 30$ et 50. Le nombre maximum d'évaluations est $nb_{eval} = 10^4 D$.

Les paramètres de la méthode NN-LCC sont $N = 100D$, $k = N/2$ et $p = 10D$. De la même manière que pour l'algorithme ABC-Morris, pour mettre en évidence l'apport de la méthode d'analyse de sensibilité, deux tests ont été réalisés : le premier, avec 100% de variables explicites et le second, en désactivant 75% des variables.

3.4.2 Résultats

Les résultats (valeurs médianes) sont présentés dans les tableaux 3.2 à 3.5. Pour chaque fonction, la meilleure valeur médiane est présentée en gras. Les valeurs marquées d'une étoile désignent les cas où la version de la métaheuristique contenant la méthode NN-LCC obtient un meilleur résultat que

la version d'origine.

Appliqué au *benchmark* CEC 2013, l'algorithme ABC-NN-LCC fournit des résultats équivalents ou meilleurs que ABCMR dans 64% des cas, en dimension 30 (Tableau 3.2) et 71% des cas, en dimension 50 (Tableau 3.3). L'algorithme DE-NN-LCC, quant à lui, fournit des résultats équivalents ou meilleurs que l'algorithme DE pour 79% des cas, en dimension 30 et 75% des cas, en dimension 50.

Dans le cas de test où 25% des dimensions ont une influence significative sur la fonction objectif (Tableaux 3.4 et 3.5), l'algorithme ABC-NN-LCC donne des résultats équivalents ou meilleurs dans 61% des cas, en dimension 30 et 61% des cas, en dimension 50. Les performances de l'algorithme initial ne sont que peu améliorés mais surtout, les résultats fournis ne sont pas discriminants par rapport à ceux observés avec 100% de dimensions actives. La trop rapide convergence de l'algorithme et sa difficulté à s'extraire d'un optimum expliquent ce comportement. L'ajout de la méthode est particulièrement bénéfique sur l'algorithme d'évolution différentielle. En effet, en dimension 30, l'algorithme DE-NN-LCC améliore l'algorithme original dans 82% des cas et dans 96% des cas, en dimension 50.

L'algorithme qABCMorris fournit de meilleurs résultats sur les fonctions sur lesquelles aucune rotation n'est appliquée. Comme nous l'avons vu pour l'algorithme précédent, ce comportement est encore vérifié ici. C'est un résultat attendu, car la méthode de Morris évalue plus précisément les dimensions inactives. Néanmoins, la méthode est beaucoup plus « résistante » aux rotations. La surestimation de certains poids permet d'effectuer une recherche dans la dimension correspondante, alors que la méthode de Morris l'évite.

Les résultats obtenus montrent que l'ajout de la méthode d'analyse de sensibilité NN-LCC dans une métaheuristique ne dégrade pas ses performances et au contraire l'améliore globalement. Néanmoins la performance de l'algorithme ainsi obtenu dépend surtout de la performance de l'algorithme original.

Les figures 15 à 18, présentées en annexe D, illustrent les principaux types d'évolution de la valeur médiane de l'erreur pour les algorithmes DE et DE-NN-LCC en fonction du nombre d'itérations, en dimensions 30 et 50, pour 25% de dimensions actives.

Nous pouvons remarquer que, pour la majorité des fonctions, l'écart à l'avantage de la méthode NN-LCC est plus important en dimension 30 qu'en dimension 50. Cela illustre le fait que la méthode perd en précision alors que le nombre de dimensions augmente. Néanmoins, cette observation est à nuancer, car les résultats des fonctions $f8$, $f9$ et $f10$ s'améliorent en dimension 50.

Lorsqu'aucune rotation n'est appliquée, il ne se détache pas de règle générale. Pour les fonctions $f5$, $f7$, $f17$ et $f22$, l'évolution est identique en dimension 30. Seule la fonction $f11$ se voit améliorée,

TABLEAU 3.2 : Erreur médiane pour 100% de dimensions actives en 30-D

	qABCMorris	ABCMR	ABC-NN-LCC	DE	DE-NN-LCC
f1	**0,00e+00**	**0,00e+00**	**0,00e+00**	1,27e+00	4,43e-04*
f2	3,07e+07	9,63e+05	1,50e+06	**3,55e+05**	3,61e+05
f3	5,88e+10	4,23e+07	2,48e+07*	3,97e+07	**1,47e+07***
f4	8,50e+04	6,92e+04	6,85e+04*	**1,09e+03**	1,41e+03
f5	**0,00e+00**	**0,00e+00**	**0,00e+00**	9,59e+01	1,96e-01*
f6	**1,62e+01**	1,67e+01	1,69e+01	2,81e+01	1,73e+01*
f7	3,18e+02	7,93e+01	7,89e+01*	3,08e+01	**1,26e+01***
f8	2,10e+01	**2,09e+01**	2,10e+01	2,10e+01	2,10e+01
f9	3,04e+01	2,18e+01	2,08e+01*	1,95e+01	**1,70e+01***
f10	1,92e-01	5,91e-02	**4,47e-02***	2,67e+00	2,29e-01*
f11	**0,00e+00**	6,47e+01	3,92e+01*	2,90e+01	2,39e+01*
f12	3,11e+02	1,64e+02	1,15e+02*	**4,60e+01**	9,60e+01
f13	3,26e+02	2,19e+02	1,77e+02*	**1,55e+02**	1,58e+02
f14	**1,87e-01**	2,02e+03	2,70e+03	1,13e+03	8,94e+02*
f15	3,78e+03	**3,11e+03**	3,17e+03	7,05e+03	7,14e+03
f16	1,01e+00	2,26e-01	**2,20e-01***	2,55e+00	2,55e+00
f17	**3,04e+01**	8,08e+01	8,98e+01	6,24e+01	6,02e+01*
f18	**3,00e+01**	7,45e+01	7,55e+01	6,27e+01	5,78e+01*
f19	**5,31e-01**	5,39e+00	4,34e+00*	6,15e+00	4,64e+00*
f20	1,21e+01	1,21e+01	1,22e+01	1,20e+01	**1,17e+01***
f21	**2,00e+02**	2,21e+02	2,27e+02	3,41e+02	3,27e+02*
f22	**9,95e+00**	2,62e+03	3,88e+03	1,10e+03	9,54e+02*
f23	4,85e+03	4,10e+03	**4,07e+03***	7,08e+03	7,15e+03
f24	2,90e+02	2,56e+02	2,51e+02*	2,47e+02	**2,46e+02***
f25	3,07e+02	2,84e+02	2,78e+02*	2,65e+02	**2,64e+02***
f26	**2,00e+02**	**2,00e+02**	**2,00e+02**	3,33e+02	3,14e+02*
f27	**4,00e+02**	8,59e+02	7,86e+02*	7,00e+02	6,71e+02*
f28	3,05e+02	**3,00e+02**	**3,00e+02**	3,04e+02	**3,00e+02***

TABLEAU 3.3 : Erreur médiane pour 100% de dimensions actives en 50-D

	qABCMorris	ABCMR	ABC-NN-LCC	DE	DE-NN-LCC
f1	**0,00e+00**	**0,00e+00**	**0,00e+00**	7,52e+01	1,19e+01*
f2	3,07e+08	1,88e+06	2,11e+06	8,39e+05	**8,29e+05***
f3	2,01e+11	**5,03e+07**	7,15e+07	1,01e+08	1,05e+08
f4	1,63e+05	9,80e+04	9,42e+04*	**1,28e+03**	1,43e+03
f5	**0,00e+00**	**0,00e+00**	**0,00e+00**	4,56e+02	1,01e+02*
f6	**4,35e+01**	4,43e+01	4,50e+01	4,79e+01	4,70e+01*
f7	7,46e+02	9,10e+01	9,18e+01	5,70e+01	**3,78e+01***
f8	**2,11e+01**	**2,11e+01**	**2,11e+01**	2,12e+01	**2,11e+01***
f9	6,07e+01	4,14e+01	4,18e+01	4,11e+01	**3,96e+01***
f10	1,52e-01	**5,21e-02**	5,80e-02	8,82e+00	2,34e-01*
f11	**0,00e+00**	1,30e+02	1,24e+02*	1,11e+02	7,57e+01*
f12	8,14e+02	2,88e+02	1,85e+02*	**1,19e+02**	3,65e+02
f13	8,31e+02	4,82e+02	3,83e+02*	**3,72e+02**	3,76e+02
f14	**1,91e-01**	4,13e+03	5,09e+03	2,33e+03	1,48e+03*
f15	7,65e+03	**6,18e+03**	6,28e+03*	1,38e+04	1,39e+04
f16	1,38e+00	2,94e-01	**2,54e-01***	3,31e+00	3,33e+00
f17	**5,08e+01**	1,80e+02	1,76e+02*	1,56e+02	1,51e+02*
f18	**5,02e+01**	1,64e+02	1,58e+02*	1,76e+02	1,53e+02*
f19	**8,46e-01**	1,25e+01	1,09e+01*	2,96e+01	1,35e+01*
f20	2,15e+01	**2,10e+01**	**2,10e+01***	2,15e+01	2,16e+01
f21	**2,03e+02**	3,00e+02	3,00e+02	4,54e+02	4,22e+02*
f22	**1,02e+01**	5,83e+03	7,69e+03	2,30e+03	1,42e+03*
f23	9,97e+03	8,37e+03	**8,29e+03***	1,38e+04	1,37e+04*
f24	3,78e+02	3,21e+02	3,09e+02*	2,94e+02	**2,86e+02***
f25	4,17e+02	3,74e+02	3,55e+02*	3,23e+02	**3,20e+02***
f26	**2,01e+02**	**2,01e+02**	**2,01e+02***	3,85e+02	3,66e+02*
f27	1,96e+03	1,48e+03	1,42e+03*	1,17e+03	**1,03e+03***
f28	**4,00e+02**	**4,00e+02**	**4,00e+02**	4,21e+02	**4,00e+02***

TABLEAU 3.4 : Erreur médiane pour 25% de dimensions actives en 30-D

	qABCMorris	ABCMR	ABC-NN-LCC	DE	DE-NN-LCC
f1	1,02e-02	4,17e-04	5,30e-04	9,92e-03	**2,30e-05***
f2	2,20e+04	2,47e+02	1,90e+02*	1,71e+02	**3,56e+00***
f3	1,77e+07	2,87e+02	8,86e+02	5,72e+02	**1,13e+01***
f4	4,20e+02	1,43e+02	2,15e+02	**3,20e+00**	3,88e+00
f5	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**	**0,00e+00**
f6	6,00e-03	1,52e-04	**5,73e-05***	1,12e-03	1,31e-03
f7	2,35e+00	1,43e+00	7,47e-01*	9,08e-03	**6,19e-03***
f8	4,44e+00	6,48e+00	3,67e+00*	8,36e-02	**6,87e-02***
f9	7,78e-01	7,94e-01	4,92e-01*	**8,37e-02**	1,11e-01
f10	3,91e-01	2,04e-01	1,48e-01*	**1,30e-01**	1,50e-01
f11	**0,00e+00**	2,99e+00	3,98e+00	9,95e-01	**0,00e+00***
f12	1,13e-02	2,02e-02	6,47e-03*	2,70e-04	**1,79e-07***
f13	7,86e-03	2,50e-02	8,07e-03*	9,74e-05	**2,28e-07***
f14	**0,00e+00**	4,55e+01	1,79e+02	8,03e+00	5,06e+00*
f15	2,37e+02	3,54e+02	2,43e+02*	3,12e+01	**1,65e+01***
f16	2,21e-01	1,26e-01	**1,22e-01***	7,14e-01	6,46e-01*
f17	**1,30e+00**	8,25e+00	1,23e+01	7,19e+00	7,10e+00*
f18	**2,72e-01**	8,67e+00	1,11e+01	7,82e+00	7,70e+00*
f19	**1,02e-02**	4,94e-02	1,49e-01	5,03e-01	2,20e-01*
f20	5,72e-01	1,18e+00	7,97e-01*	5,37e-01	**4,50e-01***
f21	**5,24e+00**	1,00e+02	1,00e+02	2,00e+02	2,00e+02
f22	**0,00e+00**	2,28e+02	3,40e+02	2,11e+02	2,06e+02*
f23	4,37e+02	4,47e+02	3,37e+02*	3,27e+02	**2,55e+02***
f24	**1,03e+02**	1,04e+02	**1,03e+02***	**1,03e+02**	1,04e+02
f25	**1,00e+02**	**1,00e+02**	**1,00e+02**	**1,00e+02**	**1,00e+02**
f26	**1,00e+02**	**1,00e+02**	**1,00e+02**	**1,00e+02**	**1,00e+02**
f27	**1,05e+02**	3,05e+02	2,45e+02*	3,00e+02	3,00e+02
f28	**1,00e+02**	**1,00e+02**	**1,00e+02**	3,00e+02	3,00e+02

TABLEAU 3.5 : Erreur médiane pour 25% de dimensions actives en 50-D

	qABCMorris	ABCMR	ABC-NN-LCC	DE	DE-NN-LCC
f1	3,41e-02	**6,68e-03**	9,25e-03	1,26e-02	7,46e-03*
f2	8,01e+05	1,06e+04	1,14e+04	2,93e+03	**1,13e+03***
f3	5,08e+08	1,29e+02	7,34e+02	7,03e+02	**1,06e+01***
f4	2,04e+03	4,99e+02	3,82e+02*	1,36e+01	**1,05e+01***
f5	**0,00e+00**	**0,00e+00**	**0,00e+00**	4,08e-03	**0,00e+00***
f6	3,96e-03	2,15e-05	2,38e-05	4,08e-05	**1,81e-05***
f7	1,39e+01	7,09e+00	3,40e+00*	9,38e-01	**4,23e-01***
f8	1,63e+01	2,00e+01	1,97e+01*	5,58e+00	**5,48e+00***
f9	2,97e+00	1,98e+00	1,13e+00*	7,10e-01	**6,87e-01***
f10	9,85e-01	3,12e-01	3,49e-01	2,76e-01	**2,06e-01***
f11	**0,00e+00**	1,09e+01	1,29e+01	2,99e+00	9,95e-01*
f12	9,95e-02	4,16e-02	2,23e-02*	1,17e-02	**4,18e-03***
f13	8,22e-01	5,03e-01	**2,11e-01***	2,74e-01	2,50e-01*
f14	**0,00e+00**	2,50e+02	4,42e+02	1,05e+02	4,85e+01*
f15	1,01e+03	8,84e+02	7,05e+02*	5,53e+02	**2,73e+02***
f16	3,82e-01	2,25e-01	**1,41e-01***	1,18e+00	1,20e+00
f17	**1,22e+01**	2,34e+01	2,67e+01	1,41e+01	1,24e+01*
f18	**1,26e+01**	2,31e+01	2,40e+01	1,52e+01	1,34e+01*
f19	**4,31e-02**	5,56e-01	8,04e-01	9,53e-01	9,11e-01*
f20	2,61e+00	2,46e+00	2,31e+00*	2,00e+00	**1,82e+00***
f21	**1,13e+02**	**2,00e+02**	**2,00e+02**	**2,00e+02**	**2,00e+02**
f22	**4,22e-04**	3,18e+02	5,65e+02	1,62e+02	1,42e+02*
f23	1,08e+03	9,97e+02	7,18e+02*	5,45e+02	**2,25e+02***
f24	**1,01e+02**	**1,01e+02**	**1,01e+02**	**1,01e+02**	**1,01e+02**
f25	**1,00e+02**	**1,00e+02**	**1,00e+02**	**1,00e+02**	**1,00e+02**
f26	**1,00e+02**	**1,00e+02**	**1,00e+02**	**1,00e+02**	**1,00e+02**
f27	3,80e+02	3,64e+02	**3,41e+02***	3,51e+02	**3,41e+02***
f28	**1,00e+02**	**1,00e+02**	**1,00e+02**	4,00e+02	4,00e+02

pour l'algorithme DE, elle doit être bloquée sur un optimum local. En dimension 50, pour ces mêmes fonctions, des évolutions similaires sont observables pour les fonctions $f7$, $f11$ et $f22$, où les deux algorithmes stagnent sur un optimum local, DE-NN-LCC atteignant une valeur légèrement meilleure. Pour la fonction $f5$, la méthode NN-LCC permet de trouver une valeur optimale.

Pour les fonctions composées ($f20$-$f28$), représentées par les graphiques $f20$, $f22$ et $f27$, le comportement des deux algorithmes est similaire ; en dimension 50, une légère amélioration est notable.

Il n'y a globalement aucune amélioration notable au-delà de 20000 itérations, pour les deux méthodes. Leur comportement de convergence est similaire, la méthode NN-LCC aura permis à l'algorithme DE de converger vers une meilleure solution, mais elle ne modifie pas son comportement de convergence prématurée.

3.5 Conclusion

Dans ce chapitre, nous avons présenté une méthode d'analyse de sensibilité simple qui permet d'acquérir de la connaissance sur un modèle, ré-injectable dans le processus de recherche d'une métaheuristique.

Les résultats sont prometteurs. En effet, lorsque toutes les variables d'entrée ont une influence significative sur la fonction objectif, la méthode évalue correctement chaque influence ainsi que leurs rangs. Dans le cas où certaines variables d'entrée ont une influence négligeable, la méthode surestime les variables de moindre poids et sous-estime les plus influentes. Néanmoins, malgré ce lissage observé des poids, elle préserve les rangs entre les variables, gardant l'étude cohérente et la priorisation correcte.

Cela nous permet alors d'inclure cette méthode dans le processus de recherche des métaheuristiques qui utilisent un déplacement le long de plusieurs dimensions simultanément. Comme précédemment avec la méthode de Morris, la méthode NN-LCC utilise les évaluations effectuées par la métaheuristique afin de construire un vecteur de poids. Ce vecteur est ensuite intégré dans l'équation de génération d'un nouvel individu, afin d'explorer en priorité les dimensions les plus prometteuses. Une nouvelle fois, pour guider le processus de recherche, l'intégration d'une méthode d'analyse de sensibilité dans une métaheuristique donne de meilleurs résultats qu'une simple recherche aléatoire.

Même si la méthode construite ici, contrairement à la méthode de Morris, ne peut affecter un poids nul à un ensemble de variables, elle est profitable aux métaheuristiques à déplacement multidimensionnel, comme la méthode de Morris l'est pour un algorithme à déplacement unidimensionnel. De plus, la surestimation du poids des variables de faible influence reste un avantage, dans le contexte de l'utilisation au sein d'une métaheuristique. En effet, au cours du processus de recherche, nous ne souhaitons pas systématiquement ignorer ces dimensions. L'objectif est de sélectionner prioritairement

les dimensions les plus influentes, afin de converger plus rapidement. La préservation des rangs dans l'évaluation des poids permet ce comportement.

La performance de l'algorithme ainsi défini dépend fortement de celle de la métaheuristique initiale. Aussi, l'amélioration fournie par l'algorithme ainsi construit se constate sur les *benchmarks* effectués.

Chapitre 4

Application à l'utilisation de méthodes inverses pour la caractérisation de matériaux à changement de phase

Ce chapitre décrit les travaux en cours qui font suite à la thèse de William Maréchal [Maréchal, Septembre 2014] : « Utilisation de méthodes inverses pour la caractérisation de matériaux à changement de phase ». Il présente le contexte de l'application, définit le problème d'optimisation et décrit les questions soulevées par l'implémentation.

Notions abordées dans ce chapitre :

- *Matériaux à changement de phase ;*
- *Calorimétrie différentielle à balayage ;*
- *Méthodes d'inversion ;*
- *Métaheuristiques et analyse de sensibilité.*

4.1 Présentation

La conscience du réchauffement climatique et du caractère polluant des énergies fossiles conduit à se donner pour objectif de maîtriser la consommation d'énergie. Le secteur du bâtiment (résidentiel et tertiaire) consomme près de la moitié de l'énergie finale nationale et produit 23,76% des émissions de CO_2 [Commissariat général au développement durable, 2016]. Le chauffage en hiver en est la principale cause.

4.1.1 Contexte

Le confort thermique est lié à plusieurs facteurs. Il dépend du niveau de température, mais aussi de sa régularité. L'inertie thermique d'un matériau caractérise sa résistance aux changements de température. Plus celle-ci est élevée, mieux le matériau résiste aux variations thermiques.

L'augmentation de l'inertie thermique d'un bâtiment permet de le rendre plus résistant aux variations de température et de compenser en partie leurs effets. La solution la plus répandue actuellement est l'augmentation de l'épaisseur des murs, par superposition de couches de matériaux différents, ce qui entraîne une perte d'espace. La réduction des besoins en énergie d'un bâtiment peut être obtenue par l'utilisation de matériaux performants pour les enveloppes du bâtiment. Ces matériaux, dits *Matériaux à Changement de Phase* (MCP) [Baetens et collab., 2010; Nkwetta et Haghighat, 2014] stockent ou libèrent de l'énergie, permettant d'amortir les variations de température au sein d'un bâtiment. Ils possèdent une densité importante de stockage et permettent de fixer le niveau de température par changement d'état. Le fabriquant BASF annonce que sa plaque *micronal® PCM smartboard*^{TM} de 1,5 cm d'épaisseur possède la même capacité d'accumulation de chaleur qu'un mur de béton de 9 cm ou qu'un mur de briques de 12 cm.

Le principe est simple : lorsque l'environnement atteint une certaine température (propre au matériau), les MCP se liquéfient en absorbant de l'énergie. Lorsque la température redescend en dessous de cette valeur, ils libèrent l'énergie en se solidifiant. Ces matériaux sont organiques (paraffine, acides gras) ou bien non organiques (sel hydraté) et sont mélangés aux matériaux de construction (béton, plâtre).

Afin de choisir précisément le MCP à utiliser, il est nécessaire de pouvoir modéliser, d'une part, le comportement énergétique d'un bâtiment, et d'autre part le processus du changement de phase. Une connaissance précise des propriétés thermophysiques du matériau est par ailleurs impérative [Dutil et collab., 2014], pour le caractériser, l'évaluer et le certifier commercialement.

Il convient alors d'établir le bilan énergétique du matériau. Une description détaillée de la résolution mathématique du problème peut être trouvée dans la thèse de William Maréchal [Maréchal, Septembre

2014]. Dans le cadre d'un matériau isotrope, le bilan de puissance s'écrit :

$$\iiint_V \left(\rho \frac{\partial h(T)}{\partial t} - \overrightarrow{\bigtriangledown}.(\lambda \overrightarrow{\bigtriangledown}(T)) \right) dV = 0. \tag{4.1}$$

où V est le volume, T la température, t le temps, ρ la masse volumique et λ un scalaire représentant le tenseur de conductivité thermique ; $\overrightarrow{\bigtriangledown}.$ représente l'opérateur de divergence et h, l'enthalpie massique du matériau (i.e. l'énergie absorbée lors du changement de phase, par exemple le passage de l'état solide à l'état liquide).

L'intégrale est nulle quel que soit le volume. Considérant la masse volumique constante, le bilan de puissance s'exprime à un niveau local selon (4.2) :

$$\rho \frac{\partial h(T)}{\partial t} = \overrightarrow{\bigtriangledown}.(\lambda \overrightarrow{\bigtriangledown}(T)). \tag{4.2}$$

Pour résoudre cette équation en T, on utilise une équation d'état qui caractérise le matériau. C'est dans cette équation qu'est pris en compte le changement de phase. L'objectif est de suivre l'évolution de l'enthalpie massique selon la température $h(T)$, afin d'étudier les propriétés énergétiques du matériau. Les figures 4.1 et 4.2 illustrent l'évolution de $h(T)$ pour un corps pur et pour une solution binaire. On peut remarquer que, dans le cas d'une solution binaire, la température de fusion T_M est inférieure à celle de l'eau pure, T_A, et que le processus de fusion s'étale sur plusieurs degrés de température. Les grandeurs C_L et C_S représentent les capacités calorifiques à l'état liquide et à l'état solide (i.e. la quantité de chaleur à fournir pour élever sa température).

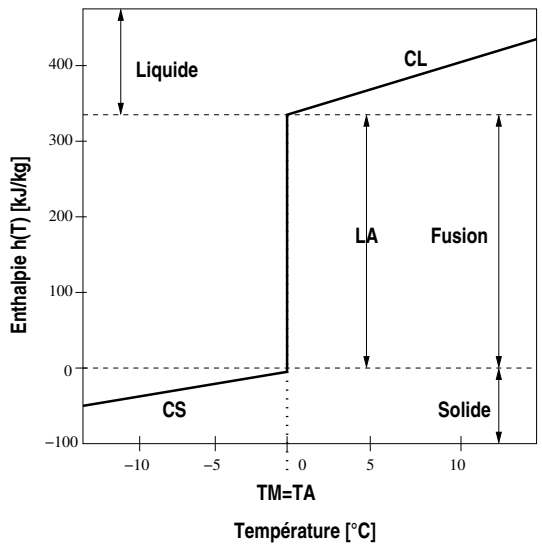

FIGURE 4.1 : Enthalpie d'un corps pur (eau).

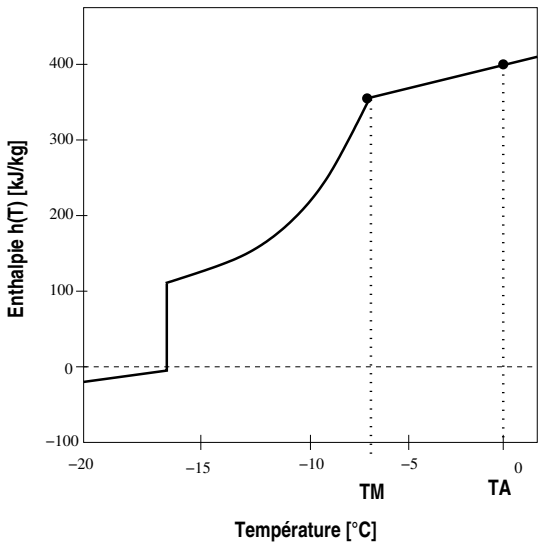

FIGURE 4.2 : Enthalpie d'une solution binaire (H_2O-NH_4Cl).

4.1.2 Calorimétrie différentielle à balayage

La calorimétrie différentielle à balayage (*Differential Scanning Calorimetry*, DSC) par compensation de puissance est un dispositif expérimental permettant de déterminer les propriétés énergétiques d'un matériau. Elle caractérise la transformation du matériau, due à la variation d'un paramètre du milieu (dite changement de phase). La mesure effectuée est la variation (i.e. la dérivée) de l'enthalpie (énergie totale d'un système thermodynamique) en fonction de la variation de température.

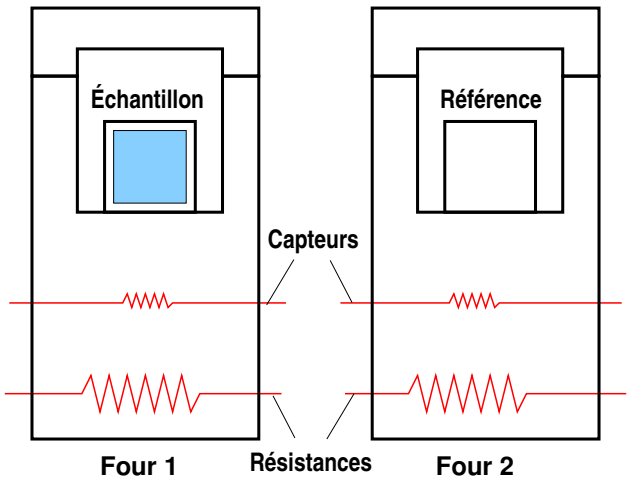

FIGURE 4.3 : Schéma d'un calorimètre.

Un calorimètre (Figure 4.3, [Maréchal, Septembre 2014]) va mesurer le flux thermique d'un échantillon. Ce flux représente l'énergie échangée entre un échantillon et une référence, lors de la variation de température. La cellule de référence est une cellule laissée vide, la seconde cellule contient, quant à elle, l'échantillon à étudier.

L'échange thermique est réalisé par un courant d'azote ou d'hélium sec. Les capteurs permettent de contrôler que l'évolution de température par les résistances électriques est équivalente dans les deux fours. La différence de puissance électrique entre les résistances est le flux thermique mesuré. Le résultat est un thermogramme (Figure 4.4). Il donne l'évolution du flux thermique ϕ en fonction du temps t ou de la température T.

La caractérisation réalisée à partir des thermogrammes d'une DSC est critiquée : ces derniers dépendent de la masse de l'échantillon, de la vitesse de montée en température, ils expriment mal les conditions aux limites. De plus, l'interprétation du thermogramme pour déduire l'enthalpie conduit à une erreur. En effet, comme le montrent les auteurs dans [Franquet et collab., 2012], si l'on considère le thermogramme pour l'eau, en fonction de la température (Figure 4.4) avec la vitesse de réchauffement, $\beta = 5\ K.min^{-1}$, la déduction de son enthalpie par intégration conduit à une erreur illustrée par la figure 4.5.

Pour éviter plusieurs expériences avec différents appareils et pour déterminer de manière correcte

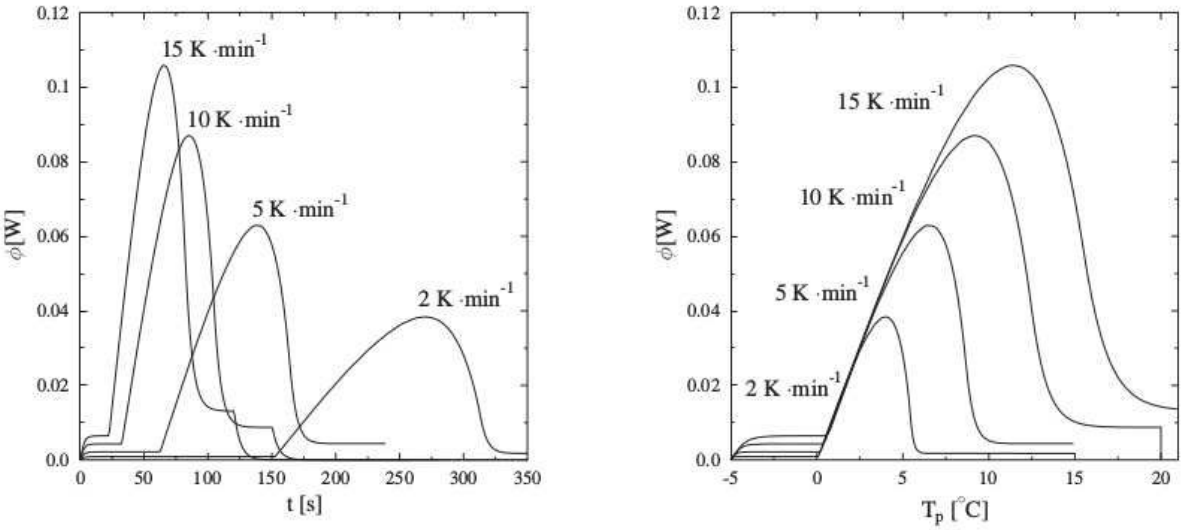

FIGURE 4.4 : Exemples de thermogrammes d'analyse de l'eau, par rapport au temps (à gauche), à la température (à droite) [Maréchal, Septembre 2014].

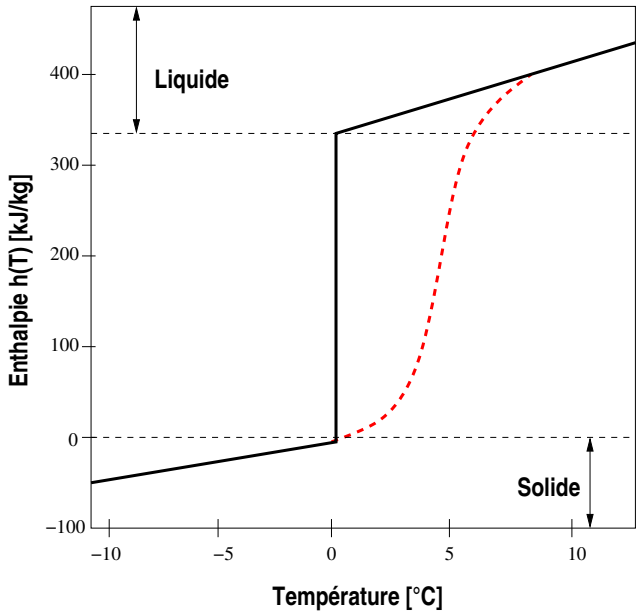

FIGURE 4.5 : Profils d'enthalpie pour l'eau : exact (noir) et déduit par intégration du thermogramme (tirets rouges)), $\beta = 5\ K.min^{-1}$.

l'enthalpie, un modèle mathématique et une géométrie décrivant le fonctionnement d'un calorimètre ont alors été construits et leur résolution a été proposée dans [Maréchal, Septembre 2014]. Le modèle ainsi développé simule l'évolution thermique d'un matériau en changement de phase solide-liquide, équivalent à une mesure DSC. Ce modèle considère l'échantillon comme une géométrie en deux dimensions, un cylindre, possédant trois caractéristiques différentes en chacune de ses faces.

Un modèle numérique réduit, exprimant l'enthalpie en fonction de la température (cf éq. (4.3)),

a ensuite été développé et validé [Gibout et collab., 2013]. Ce dernier regroupe les équations de conservation, les conditions aux limites et l'équation d'état. Il fournit de manière rapide et fiable une caractérisation approchée de l'enthalpie d'un échantillon, en s'abstrayant notamment de la géométrie de l'échantillon, considérant celle-ci comme une sphère possédant une unique caractéristique en tout point de sa surface.

$$\frac{dh}{dT} = \begin{cases} \frac{T_A - T_M}{(T_A - T)^2} L_A + C_S \left(1 - \frac{T_A - T_M}{T_A - T}\right) + C_L \left(\frac{T_A - T_M}{T_A - T}\right) & \text{pour } T < T_M, \\ \\ C_L & \textit{sinon.} \end{cases} \tag{4.3}$$

où C_S et C_L représentent respectivement les capacités calorifiques aux états solide et liquide, L_A est la chaleur latente de fusion, T_A et T_M sont les températures de fusion du corps pur et du *liquidus* (solution binaire).

Nous avons donc à notre disposition un code de calcul qui modélise l'évolution du flux de conductivité.

4.2 Le problème d'optimisation : méthodes d'inversion pour la calorimétrie

Un problème inverse consiste à déterminer les grandeurs physiques, causes d'un phénomène observable, à partir du résultat d'une expérience.

Il nécessite la modélisation mathématique $\mathcal{M}$ de l'expérience dont on cherche à déterminer l'ensemble de paramètres, noté $X^* = {}^T(x_1^*, ..., x_D^*)$. L'expérience a conduit au résultat observé y_{obs}. Une simulation de l'expérience est l'application du modèle $\mathcal{M}$, à un ensemble de variables X, fournissant une solution $y = \mathcal{M}(X)$.

Résoudre un tel problème consiste à minimiser l'écart entre une solution simulée y et la valeur observée y_{obs}. La fonction objectif est donc une norme, adéquate au problème, pour calculer cet écart.

Le problème d'optimisation est l'utilisation d'une méthode inverse pour identifier les paramètres d'une expérience de DSC [Franquet et collab., 2012]. Ces paramètres sont les propriétés de l'échantillon décrites ci-après.

4.3 Mise en place de la solution

Nous avons défini le problème d'optimisation et pouvons maintenant décrire les variables et les paramètres du problème.

4.3.1 Variables et paramètres

Les variables ($X = {}^T(x_1, ..., x_D)$) du problème d'optimisation sont composées des paramètres de l'équation d'état ainsi que des paramètres de l'échantillon. Elles sont décrites dans le tableau 4.1. La colonne « Valeur d'expérience » regroupe les valeurs des paramètres qui ont permis de générer la valeur d'observation y_{obs}.

TABLEAU 4.1 : Variables de l'expérience.

	Signification	Unité	Min	Max	Valeur d'expérience
Paramètres de l'équation d'état du matériau					
C_S	Capacité calorifique état solide	$J.K^{-1}$	100	1e+04	2000
C_L	Capacité calorifique état liquide	$J.K^{-1}$	100	1e+04	4200
T_A	Température de fusion du corps pur	°C	T_{deb}	T_{fin}	15
T_M	Température de fusion du *liquidus*	°C	T_{deb}	T_A	14,99
L_A	Chaleur latente de fusion	$J.kg^{-1}$	0	5e+05	1,2e+05
Paramètres de l'échantillon					
K_L	Conductivité thermique à l'état liquide	$W.m^{-1}.K^{-1}$	1e-03	3	1
K_S	Conductivité thermique à l'état solide	$W.m^{-1}.K^{-1}$	1e-03	3	0,5
R_1	Résistance thermique, face intérieure	$K.W^{-1}$	1e-04	1e-01	5e-03
R_2	Résistance thermique, face latérale	$K.W^{-1}$	1e-04	1e-01	5e-03
R_3	Résistance thermique, face supérieure	$K.W^{-1}$	1e-04	1e-01	2e-02

L'environnement de l'expérience est régi par certains paramètres (cf Tableau 4.2) qui ne font pas partie du problème d'optimisation. Ils servent uniquement au réglage de l'expérience et sont injectés tels quels dans le modèle mathématique.

TABLEAU 4.2 : Paramètres de l'expérience.

Paramètre	Signification	Unité	Valeur d'expérience
T_{deb}	Température initiale	°C	5
T_{fin}	Température finale	°C	20
β	Vitesse de réchauffement	$K.min^{-1}$	3.33e-02
ρ	Masse volumique	$kg.m^{-3}$	1e+04
d	Durée	s	600
pas_{log}	Pas de temps de sauvegarde des données	s	1
dt	Pas de temps de résolution du modèle	s	0,1

4.3.2 Fonction objectif et solution

La fonction objectif à minimiser mesure l'erreur quadratique entre le résultat du modèle mathématique et la donnée observée. Elle est donnée par l'équation (4.4) :

$$F_{obj}(X) = \sum_{i=1}^{n} (\phi(t_i, X) - \phi_{obs}(t_i))^2. \tag{4.4}$$

```
 0.0 5.0 -6.016604629477566E-18
1.0000000000000007 5.033333333333333 -2.0816876560652225E-4
2.0000000000000013 5.066666666666666 -3.9461836657418676E-4
3.00999999999998 5.1003333333333325 -5.65576089774739E-4
...
597.0099999996021 20.0 -8.429987165983157E-14
598.0099999996012 20.0 -7.058067249780415E-14
599.0099999996003 20.0 -5.909902240892416E-14
```

FIGURE 4.6 : Résultat de référence de l'expérience (temps, température, valeur de flux).

où $n = d/pas_{log}$ correspond au nombre d'enregistrements effectués, $\phi(t_i, X)$ est la valeur de flux pour le jeu de paramètres X au temps t_i et $\phi_{obs}(t_i)$ est la valeur du flux de référence observé par l'expérience de DSC. n La sortie du modèle mathématique est constituée de différentes informations :

— le i^e pas de temps t_i, d'enregistrement de la mesure ;

— la température $T(t_i)$, à ce pas de temps ;

— le flux du thermogramme $\phi(t_i)$, à cet instant.

Le vecteur de flux de référence ϕ_{obs} est un paramètre de l'algorithme. La figure 4.6 illustre les premières et les dernières lignes de l'observation de référence. Les valeurs de flux de chaque simulation effectuée par le code de calcul sont prises aux mêmes intervalles de temps que la référence.

Une solution X_i du problème d'optimisation est une configuration des variables décrites dans le tableau 4.1. La figure 4.7 résume le procédé de la méthode inverse.

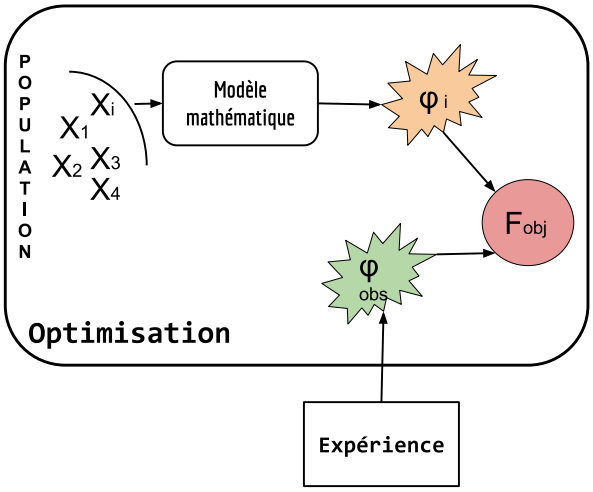

FIGURE 4.7 : Processus de fonctionnement du problème d'optimisation.

4.3.3 Métaheuristique

Les travaux précédents [Maréchal, Septembre 2014] ont utilisé une version modifiée de la méthode de Levendberg-Marquardt [Ranganathan, 2004] (méthode adaptative entre la descente de gradient

et la méthode de Gauss-Newton) et la méthode du simplexe. Dans ces travaux, le nombre de paramètres considérés en tant que variables du problème d'optimisation est réduit et, dans ce contexte, les méthodes de descente et de Levendberg-Marquardt sont rapides et fiables.

L'objectif de notre application est de vérifier si une métaheuristique pourrait donner de bons résultats pour un nombre plus élevé de paramètres laissés en variables. L'utilisation de métaheuristiques, notamment les algorithmes génétiques pour ce type de problème [Gosselin et collab., 2009], est courante. Cependant, des considérations physiques laissent à penser que plusieurs variables n'auraient que peu d'influence sur le calcul du flux de conductivité et par conséquent, sur le résultat de la fonction objectif. De plus, le modèle numérique est très onéreux en temps de calcul, une convergence rapide serait donc profitable. Dans ce contexte, il parait pertinent d'utiliser un algorithme d'optimisation couplé à une méthode d'analyse de sensibilité, tel que présenté précédemment, afin de réduire le nombre d'appels au modèle, tout en garantissant la convergence.

Nous utilisons les algorithmes GABC/GABC-Morris et DE/DE-NN-LCC avec les schémas de mutation DE/best/1 (1.7), DE/current-to-best/2 (1.10) et DE/rand/2 (1.6). L'objectif est de comparer les résultats entre une métaheuristique et sa version avec analyse de sensibilité, afin de constater une amélioration de la convergence. Aussi, nous souhaitons comparer les performances des méthodes d'analyse de sensibilité entre elles. Les versions des algorithmes ABC et DE impliquant le meilleur individu de la population ont été retenues, compte tenu de la faisabilité / non faisabilité des solutions abordée ci-après.

La métaheuristique choisie va faire évoluer les variables définies Tableau 4.1. Chaque évaluation fournit un vecteur de flux. L'erreur quadratique moyenne entre ce flux et le flux de référence est l'objet de la minimisation. En fin d'exécution de l'algorithme, on obtient une courbe calant celle du flux de référence. On obtient ainsi un vecteur de paramètres pour l'équation d'état et pour l'échantillon (Tableau 4.3). Pour ces paramètres, le flux d'échange de chaleur, fonction de la température, est illustré par la figure 4.8.

TABLEAU 4.3 : Exemple de résultats des paramètres fournis par l'optimisation (pour 1000 évaluations du modèle).

	C_S	C_L	T_A	T_M	L_A
Référence	2000	4200	15	14,99	1,2e+05
GABC	7238,58	10000	18,40	18,39	0
DE/best/1	100	666,67	19,91	18,68	2,94+e05

	K_L	K_S	R_1	R_2	R_3
Référence	1	0,5	5e-03	5e-03	2e-02
GABC	2,38	2,59	1,01e-04	1,01e-04	4,58e-03
DE/best/1	1,11e+03	8,43e+02	2,11e-04	2,11e-04	1,9e-04

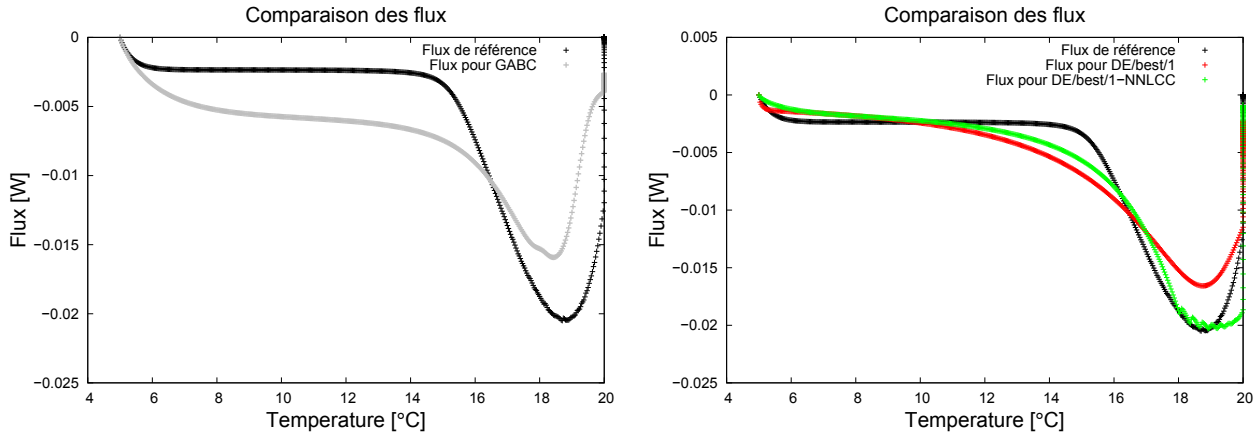

FIGURE 4.8 : Exemple de comparaison des flux de référence et d'optimisation.

4.4 Discussion

4.4.1 Dépendance entre variables

Le lien entre les variables T_A et T_M présente plusieurs contraintes. Tout d'abord, la température de fusion T_M du *liquidus* est inférieure ou égale à celle du corps pur T_A. Différentes stratégies de traitement de la dépendance entre ces variables sont envisagées. Parmi ces options, il est important de tenir compte du fait que ce lien peut rendre impossible l'utilisation de l'algorithme ABC-Morris. En effet, la méthode de Morris ne s'applique pas si deux paramètres varient simultanément. C'est une contrainte à prendre en compte dans l'élaboration de la stratégie.

Faisabilité de la solution : Si une solution est engendrée avec $T_M > T_A$, c'est une solution infaisable, traitée en tant que telle, selon la méthode choisie.

Bornage : Si $T_M > T_A$, la solution sort de l'espace de recherche et on borne les valeurs (on les fixe égales ou on fait « rebondir » T_M sur T_A).

Pas de décalage : Pour éviter une gestion de deux paramètres contraints, on ne traite pas la variable T_M comme une température variable. On la remplace par un delta de différence. La valeur effective de T_M sera alors calculée par soustraction de ce delta à T_A. Cette méthode a pour avantage de ne jamais inverser les deux températures. Le bornage de T_A va essentiellement concerner le non dépassement des bornes minimale et maximale de température.

4.4.2 Divergence du modèle

Sous certaines configurations de variables d'entrée, le modèle diverge et ne fournit pas de réponse. Ceci est dû en partie à une valeur trop grande du pas de temps de résolution dt pour le paramétrage envoyé au modèle. Un moyen de résoudre ce problème est de diminuer progressivement ce pas de temps pour que le modèle fournisse une réponse. Cette diminution progressive du pas de temps rend

l'exécution du modèle extrêmement coûteuse en temps (de quelques minutes à plus d'une journée pour un millier d'évaluations).

Afin de conserver des temps d'exécution raisonnables, une solution est de considérer que le problème d'optimisation est un problème d'optimisation sous contrainte, sans explicitation de la contrainte. Si une solution est infaisable, le code de calcul du modèle mathématique provoque une erreur et aucune évaluation n'est fournie. Ce comportement particulier, ainsi que la non connaissance de la contrainte, rendent difficilement applicables les stratégies usuelles de l'optimisation sous contrainte.

La réinitialisation d'une solution jusqu'à l'obtention d'une solution faisable n'est pas envisageable, l'exécution du modèle étant trop coûteuse. La comparaison de deux solutions divergentes n'est pas possible, car l'erreur du code de calcul n'est pas quantifiable. La pénalisation d'une telle solution reste néanmoins envisagée. Nous avons retenu la pénalisation « mortelle » de la solution (i.e. l'affectation de la valeur $+\infty$ à l'évaluation, dans le cas d'un problème de minimisation). Étant donné l'impossibilité de quantifier la non faisabilité, nous ne pouvons pas définir de fonction de pénalisation ; une constante élevée est alors affectée à l'évaluation de la solution.

Le fait d'utiliser les algorithmes GABC et la mutation DE/best/1/bin, permettent d'attirer la solution non faisable vers la meilleure de la population et ainsi de la rendre à nouveau faisable. Afin de ne pas faire sortir toutes les solutions hors du domaine de faisabilité, un autre moyen est de ne pas intégrer une solution non faisable dans l'équation de déplacement/mutation de la métaheuristique.

Ce comportement empêche l'utilisation de la méthode de Morris. Étant donné le caractère stochastique de l'algorithme ABC, même en limitant le nombre de points, il n'est absolument pas certain que l'on puisse construire une trajectoire complète pour chaque point. Une solution envisagée est de ne retenir, pour l'analyse de Morris, que les points n'ayant pas été pénalisés. Cette solution est dépendante du nombre de points divergents : avec pour critère d'arrêt un millier d'évaluations, le nombre d'évaluations divergentes est compris entre 10% et 75%. Cela ne garantit pas un nombre suffisant d'évaluations pour s'assurer qu'une trajectoire a été complétée pour seulement dix points (dix étant la dimension du problème).

La diminution du pas de temps pour une solution infaisable, avant pénalisation, une seule fois, n'est pas une bonne stratégie, car cela ralentit l'exécution de l'algorithme, sans garantie de résultat. En effet, le nombre d'évaluations non divergentes n'augmente que peu pour une division par deux du pas de temps. Un nouveau problème à résoudre se présente alors : « Existe-t-il une valeur acceptable pour que ce pas de temps rende toute solution faisable ? ».

Cette problématique impacte la méthode NN-LCC dans une moindre mesure. En effet, soit on stocke, dans l'archive servant à l'analyse, uniquement les solutions faisables, soit il suffit de s'assurer qu'il existe suffisamment de solutions faisables dans l'archive et qu'un centre de voisinage n'est pas

une solution faisable.

4.5 Conclusion

Nous avons présenté dans ce chapitre la méthode inverse de résolution, appliquée à la caractérisation des MCP. La méthode inverse définit un problème d'optimisation, qui peut être résolu par les méthodes de gradient, du simplexe ou encore par l'utilisation de métaheuristiques.

Ici, le problème contient des variables qui ont peu ou pas d'influence sur l'expérience physique. L'objectif de cette application est que notre méthode les détecte, afin de concentrer la recherche sur les autres variables.

Les travaux d'adaptation de nos méthodes au problème de caractérisation des MCP sont en cours. La faisabilité d'un ensemble de paramètres du modèle mathématique posé contraint l'utilisation d'une méthode d'analyse de sensibilité. La méthode d'analyse de sensibilité ne qualifie pas précisément les influences des dimensions car, lorsqu'une simulation diverge, son évaluation est pénalisée à une valeur constante élevée (afin qu'elle n'interfère pas avec des valeurs d'écart probables).

Par conséquent, le calcul des effets élémentaires est faussé et ne produit pas d'information correctement évaluée pour chacune des dimensions. Lorsque ce cas survient, une piste explorée est de ne pas réaliser la mise à jour des effets élémentaires. Il n'y a alors aucune garantie que tous les paramètres aient eu une évaluation convenable de leur sensibilité. La méthode NN-LCC est plus facilement adaptable, néanmoins la question de son paramétrage convenable (nombre de voisinages, individus par voisinage), conduisant à une évaluation correcte des influences, reste ouverte.

Il existe encore d'autres d'incertitudes quant au paramétrage de la méthode. Les choix faits pour la gestion de la température T_M, le bornage des paramètres lorsqu'ils sortent de l'espace de recherche, ainsi que la gestion de la divergence de l'expérience (faisabilité d'une solution), doivent être affinés et confrontés à d'autres méthodes, afin de proposer une méthodologie adaptée.

Conclusion générale et perspectives

Une étude sur les métaheuristiques populaires, basées sur des métaphores biologiques et éthologiques, a été réalisée en chapitre 1. Une attention particulière a été portée sur trois algorithmes à population, dédiés à l'optimisation à variables continues : l'évolution différentielle, l'optimisation par essaim particulaire et la colonie d'abeilles artificielles. Un ensemble de caractéristiques propres à chacun a été identifié et, pour chacun d'entre eux, un état de l'art a été proposé. Parmi les critères de choix de ces métaheuristiques ont été retenus le fait qu'elles sont, dans leur définition, dédiées à l'optimisation en variables continues et qu'elles font évoluer leur population selon trois types de déplacements : le long d'une unique dimension, le long d'un sous-ensemble de dimensions et dans toutes les dimensions.

L'axe d'amélioration des métaheuristiques proposé est l'acquisition, puis l'utilisation, d'information sur le comportement global d'une fonction le long de chacune de ses dimensions. L'originalité de l'approche ne réside pas en l'amélioration de caractéristiques d'une métaheuristique particulière, mais se concentre sur la manière de déplacer judicieusement une population d'individus selon les « contraintes du terrain », quel que soit l'algorithme utilisé.

Dans cette optique, une étude d'analyse de sensibilité de la fonction objectif permet de quantifier l'influence des paramètres de la fonction sur son résultat. Le chapitre 1 a également exposé un panorama des principales méthodes du domaine. Plusieurs méthodes d'analyse ont été présentées :

— des méthodes basées sur la corrélation linéaire entre les paramètres et la réponse de la fonction ;
— des méthodes pas-à-pas et les effets élémentaires ;
— des méthodes basées sur l'analyse de la variance, plus performantes.

Utiliser une telle analyse au sein d'une métaheuristique impose certaines contraintes : aucune connaissance sur l'expression de la fonction objectif n'est présupposée (linéarité, dérivabilité, etc.) ; les points servant à l'analyse ne dépendent pas d'un plan d'expérience prédéfini mais sont générés par le déroulement de la métaheuristique ; enfin, il est nécessaire de se servir uniquement des évaluations

fournies par la métaheuristique, sans en requérir de supplémentaire.

L'algorithme ABC-Morris présente une première application de ce concept. Nous avons décrit l'incorporation de la méthode de Morris dans la métaheuristique de colonie d'abeilles artificielles (ABC). La méthode de Morris s'adapte simplement à la métaheuristique de base, ainsi qu'à un certain nombre de ses améliorations, de par la similarité des équations sur lesquelles les deux méthodes reposent. Elle se comporte comme un élément facilitateur. Comme nous l'avons vu dans les cas de GABC et qABC, un algorithme plus performant que la métaheuristique ABC originale se voit amélioré par intégration de la méthode. Les tests sur le jeu de données CEC 2013 ont montré des résultats non dégradés et performants, dans les conditions originales de tests, puis lorsqu'un ensemble de dimensions est désactivé, une forte amélioration de la convergence est constatée en faveur de l'algorithme proposé.

Cette méthode est très efficace, mais fortement contrainte à la définition de l'équation de déplacement d'un individu, dans le cas de l'algorithme de colonie d'abeilles artificielles. En effet, elle n'est pas compatible avec certaines de ses améliorations comme l'algorithme I-ABC, de par l'expression de son équation de déplacement, ou comme l'algorithme ABC à taux de modification, car celui-ci provoque un déplacement multi-dimensionnel d'un individu.

Afin d'étendre le concept à ces algorithmes ainsi qu'à un ensemble plus important de métaheuristiques (telles que l'évolution différentielle), nous avons ensuite étudié de nouvelles méthodes d'analyse de sensibilité. La majorité des méthodes efficaces utilisées (analyses basées sur la variance) nécessitent un nombre très important de points et de calculs rendant difficile leur utilisation dans notre contexte. Nous avons proposé une nouvelle méthode d'analyse, basée sur les coefficients de corrélation linéaire : la méthode NN-LCC. Cette méthode réalise une étude locale, sur un ensemble de points proches, puis fournit des conclusions globales sur le comportement de chaque dimension, pour l'ensemble de l'espace de recherche. Lorsque cette méthode est intégrée au sein d'une métaheuristique, les tests ont montré une amélioration des résultats. Néanmoins, cette amélioration n'est pas aussi significative que pour l'algorithme précédent. En effet, lorsque le nombre de dimensions augmente, l'évaluation de l'influence est répartie sur l'ensemble des dimensions. Les variables les plus influentes sont sous-évaluées et les moins influentes légèrement sur-évaluées. Plus ce nombre de dimensions croît, moins l'écart-type de l'influence est important et, par conséquent, moins le résultat est précis.

Ce travail ouvre de nombreuses perspectives de recherche.

Une analyse plus poussée sur le paramétrage de la méthode NN-LCC reste à faire. Nous avons réalisé un ensemble de tests avec des valeurs sélectionnées par échantillonnage. Cependant, le nombre de points nécessaires à la recherche de voisins reste quelque peu coûteux et pourrait être réduit. Un

objectif serait de réduire le nombre de points nécessaires, afin de déterminer une série de valeurs dépendantes de la dimension pour les nombres de points considérés, de voisinages et de voisins ; et ce, dans le but de ne pas complexifier le paramétrage de la métaheuristique.

L'application des méthodes ABC-Morris et DE-NN-LCC au problème d'optimisation pour la caractérisation de matériaux à changement de phase introduit un nouveau périmètre d'étude : la gestion de la contrainte de faisabilité pour effectuer une analyse de sensibilité correcte des variables. Contrairement à de nombreux exemples de la littérature, le problème posé par l'application est que la contrainte de faisabilité n'est pas exprimée analytiquement et n'est pas quantifiable. Il reste alors à déterminer de nouvelles stratégies pour prendre en compte cette problématique et s'assurer de la pertinence de leurs résultats.

Par ailleurs, les méthodes ont uniquement été testées dans le cadre de l'optimisation statique. Appliquées à l'optimisation dynamique, elles devraient prendre plus de sens encore. Lorsque la fonction objectif est modifiée, les influences des dimensions peuvent alors changer ; une nouvelle étude de sensibilité permettrait de réajuster la recherche.

Enfin, l'axe d'amélioration proposé dans ce travail de thèse se concentre essentiellement sur l'importance d'une dimension dans le calcul de la fonction objectif. Nous avons quantifié cette importance vis-à-vis des autres dimensions par un pourcentage d'influence. Or, les deux méthodes d'analyse de sensibilité utilisées, qui conduisent la recherche de la métaheuristique, fournissent des informations supplémentaires. Elles sont capables de qualifier le comportement d'une variable sur son espace de recherche et/ou de détecter une interaction entre variables. Jusqu'à présent, cette information n'a pas été intégrée et peut faire l'objet d'une étude précise. Elle consisterait à adapter un pas de décalage selon le type de comportement d'une dimension (linéaire ou non linéaire). Cette procédure donnerait notamment du sens à son intégration dans la méthode d'optimisation par essaim particulaire. En effet, pour cette dernière, l'intérêt de seulement hiérarchiser l'influence des variables est limité, car la métaheuristique déplace les individus vers toutes les directions simultanément.

Annexes

A Résultats du test de Wilcoxon

Cette annexe présente les résultats détaillés du test de Wilcoxon pour chaque fonction et chaque pourcentage de variables explicites. Ce tableau permet d'identifier les différences significatives entre les variantes de l'algorithme ABC et leurs versions intégrant la méthode d'analyse de sensibilité de Morris. Les p-valeurs en gras correspondent à une différence significative en faveur de la version Morris d'un algorithme, une p-valeur en gris représente une différence significative pour la version originale, enfin, une police d'écriture normale indique qu'il n'y a aucune différence significative entre les deux versions de l'algorithme.

TABLEAU 4 : Résultats du test de Wilcoxon (p-valeur) par fonction et par pourcentage

		ABC	GABC	MeABC	qABC	Total
f1	10	**1.21e-12**	**2.20e-16**	**2.20e-16**	**2.20e-16**	
	25	**3.51e-05**	**3.77e-06**	**8.69e-04**	**1.08e-11**	
	50	**1.09e-06**	**1.53e-03**	**2.00e-03**	**5.43e-04**	
	75	1.62e-02	1.03e-01	**1.09e-03**	2.67e-01	**4.81e-05**
	100	NA	NA	NA	NA	
	Total	**6.64e-03**	1.16e-01	1.04e-01	1.41e-02	
f2	10	6.02e-01	6.13e-01	2.25e-02	3.82e-01	
	25	1.64e-04	2.22e-03	6.25e-06	4.79e-08	
	50	2.20e-16	4.50e-11	3.01e-11	3.89e-13	
	75	3.02e-11	3.02e-11	2.20e-16	3.02e-11	2.20e-16
	100	3.01e-11	3.02e-11	3.01e-11	2.20e-16	
	Total	7.78e-09	4.27e-09	3.26e-09	3.81e-09	
f3	10	1.09e-06	2.35e-15	3.89e-13	4.79e-08	
	25	3.02e-11	3.02e-11	3.02e-11	3.02e-11	
	50	3.02e-11	3.02e-11	3.02e-11	3.02e-11	
	75	2.20e-16	2.20e-16	3.02e-11	3.02e-11	2.20e-16
	100	2.20e-16	2.20e-16	2.20e-16	3.02e-11	
	Total	4.24e-14	2.20e-16	2.20e-16	2.42e-14	
f4	10	**1.03e-03**	1.06e-01	2.74e-02	**1.64e-05**	
	25	9.12e-01	1.37e-02	1.30e-01	2.48e-01	
	50	7.52e-01	7.97e-02	3.14e-01	1.38e-01	

Continue page suivante

TABLEAU 4 : Résultats du test de Wilcoxon (p-valeur) par fonction et par pourcentage

		ABC	GABC	MeABC	qABC	Total
	75	**6.25e-06**	7.41e-01	5.04e-01	**3.36e-03**	6.20e-01
	100	**2.42e-04**	2.24e-01	6.65e-01	**6.26e-05**	
	Total	2.98e-01	5.44e-01	6.85e-01	3.12e-01	
	10	NA	NA	NA	NA	
	25	NA	NA	NA	NA	
f5	50	NA	NA	NA	NA	
	75	NA	NA	NA	3.34e-01	3.18e-01
	100	NA	NA	NA	NA	
	Total	NA	NA	NA	3.21e-01	
	10	**6.00e-13**	**5.50e-03**	**1.02e-07**	**5.07e-16**	
	25	**6.80e-11**	**5.76e-06**	2.07e-02	**9.08e-12**	
f6	50	**3.89e-13**	**3.12e-04**	**2.11e-03**	**4.11e-06**	
	75	3.85e-02	6.11e-02	5.42e-01	**5.24e-03**	**8.78e-05**
	100	7.97e-01	4.58e-01	8.09e-01	3.14e-01	
	Total	**7.18e-03**	1.66e-01	1.70e-01	**7.36e-03**	
	10	**8.30e-05**	**1.99e-04**	1.25e-02	7.23e-02	
	25	1.09e-01	4.96e-02	3.90e-01	8.66e-01	
f7	50	**2.48e-12**	**1.13e-15**	**7.62e-14**	**1.57e-06**	
	75	**2.20e-16**	**2.20e-16**	**2.20e-16**	**2.20e-16**	**3.63e-13**
	100	**2.20e-16**	**2.20e-16**	**2.20e-16**	**2.20e-16**	
	Total	**1.90e-03**	**4.81e-05**	**5.12e-05**	**1.33e-03**	
	10	**2.18e-08**	**2.45e-06**	**9.54e-10**	**9.16e-08**	
	25	1.92e-01	1.55e-01	3.35e-01	2.85e-02	
f8	50	**3.02e-05**	4.40e-01	**3.36e-03**	1.23e-01	
	75	2.24e-01	2.54e-01	4.67e-01	4.79e-02	1.13e-01
	100	2.86e-01	2.19e-01	2.48e-01	3.66e-01	
	Total	3.24e-01	5.68e-01	2.78e-01	3.34e-01	
	10	**3.56e-09**	**2.45e-06**	**3.14e-07**	**6.20e-07**	
	25	7.75e-01	4.06e-01	1.91e-02	1.82e-01	
f9	50	5.92e-01	8.55e-01	5.23e-01	4.76e-01	
	75	8.20e-01	4.67e-01	5.52e-01	3.90e-01	2.85e-01
	100	4.23e-01	5.13e-01	7.19e-01	9.94e-01	
	Total	5.12e-01	5.34e-01	4.97e-01	4.93e-01	
	10	**4.11e-06**	**7.30e-03**	**2.41e-05**	**5.76e-06**	
	25	**1.54e-08**	**5.80e-11**	**3.89e-13**	**3.57e-10**	
f10	50	**5.18e-09**	**3.57e-10**	**2.23e-05**	**2.27e-04**	
	75	**4.11e-03**	**2.34e-03**	4.15e-02	1.92e-01	**7.86e-07**
	100	**5.29e-06**	**6.64e-09**	**9.56e-09**	7.52e-01	
	Total	1.01e-02	**5.90e-03**	**5.57e-03**	1.74e-02	
	10	NA	NA	NA	NA	
	25	NA	NA	NA	NA	
f11	50	NA	NA	NA	NA	
	75	NA	NA	NA	NA	NA
	100	NA	NA	NA	NA	
	Total	NA	NA	NA	NA	

Continue page suivante

TABLEAU 4 : Résultats du test de Wilcoxon (p-valeur) par fonction et par pourcentage

		ABC	GABC	MeABC	qABC	Total
	10	**7.70e-04**	5.04e-01	8.32e-01	**1.89e-03**	
	25	**4.58e-09**	**9.56e-09**	**1.09e-06**	**2.24e-06**	
f12	50	1.30e-01	4.30e-02	1.37e-02	5.51e-02	
	75	3.98e-01	7.75e-01	5.23e-01	9.01e-01	2.38e-01
	100	9.36e-01	4.23e-01	3.00e-01	5.62e-01	
	Total	5.62e-01	5.96e-01	5.62e-01	3.59e-01	
	10	**1.29e-03**	4.51e-02	**2.15e-03**	**3.96e-08**	
	25	3.74e-01	**3.32e-04**	5.71e-02	8.20e-01	
f13	50	5.82e-01	2.54e-01	7.64e-01	6.44e-01	
	75	1.00e+00	1.12e-01	9.71e-01	9.71e-01	3.74e-01
	100	6.87e-01	2.08e-01	9.24e-01	4.32e-01	
	Total	7.25e-01	5.29e-01	6.25e-01	5.59e-01	
	10	**3.13e-04**	5.57e-01	3.34e-01	3.45e-07	
	25	**1.56e-11**	**1.55e-08**	**6.04e-07**	**1.61e-11**	
f14	50	**1.69e-07**	**2.83e-10**	**1.02e-09**	**3.57e-11**	
	75	**2.69e-03**	**2.56e-06**	**8.32e-04**	**2.27e-04**	**4.91e-16**
	100	1.00e+00	1.42e-01	1.20e-02	5.33e-01	
	Total	**3.94e-06**	**1.05e-03**	**4.62e-03**	**1.09e-08**	
	10	1.61e-01	NA	NA	2.16e-02	
	25	5.58e-03	NA	9.86e-01	3.34e-01	
f15	50	**4.57e-12**	1.35e-01	3.61e-01	**3.19e-07**	
	75	**4.57e-12**	7.22e-01	1.00e+00	**5.77e-11**	**2.20e-16**
	100	**1.21e-12**	6.73e-01	**6.03e-03**	**1.21e-12**	
	Total	**2.20e-16**	5.28e-01	6.99e-01	**2.20e-16**	
	10	**3.32e-04**	**3.56e-09**	**1.75e-04**	**8.68e-06**	
	25	**6.26e-05**	**3.72e-03**	**1.31e-06**	**4.31e-03**	
f16	50	**1.70e-03**	**1.70e-03**	**9.75e-04**	1.06e-01	
	75	1.50e-01	7.97e-02	4.15e-01	4.06e-01	**5.30e-05**
	100	5.72e-01	2.03e-01	3.58e-01	4.06e-01	
	Total	5.09e-02	1.77e-02	4.39e-02	8.13e-02	
	10	**4.65e-09**	3.23e-02	3.58e-02	3.24e-07	
	25	1.73e-01	3.32e-02	8.25e-02	1.75e-06	
f17	50	**4.06e-10**	5.72e-02	3.81e-03	1.22e-10	
	75	**3.78e-10**	7.37e-02	9.40e-01	2.35e-06	**2.92e-03**
	100	4.67e-01	1.01e-02	1.81e-02	2.08e-01	
	Total	4.02e-02	1.48e-01	3.82e-01	3.08e-02	
	10	**1.24e-03**	6.51e-01	8.54e-02	**4.10e-11**	
	25	4.22e-02	2.45e-01	1.88e-01	**1.11e-03**	
f18	50	NA	7.20e-01	2.78e-03	3.34e-01	
	75	NA	1.00e+00	8.15e-02	3.34e-01	3.10e-01
	100	NA	3.34e-01	3.34e-01	NA	
	Total	4.80e-01	9.94e-01	6.68e-01	1.74e-01	
	10	**5.80e-06**	1.37e-03	1.21e-01	1.14e-02	
	25	9.82e-01	1.69e-02	**5.77e-03**	5.42e-01	
f19	50	**3.19e-03**	3.45e-02	9.06e-02	**1.43e-04**	

Continue page suivante

TABLEAU 4 : Résultats du test de Wilcoxon (p-valeur) par fonction et par pourcentage

		ABC	GABC	MeABC	qABC	Total
	75	**3.18e-06**	4.32e-01	1.50e-01	**7.38e-06**	6.70e-01
	100	**5.80e-11**	4.46e-02	**3.91e-03**	**4.29e-08**	
	Total	1.32e-01	1.70e-01	3.54e-01	3.65e-02	
	10	**8.30e-05**	**2.84e-07**	**2.27e-04**	**1.53e-03**	
	25	2.93e-01	1.73e-01	1.62e-02	4.23e-01	
f20	50	9.47e-01	**6.65e-03**	9.01e-01	2.36e-01	
	75	4.85e-01	1.64e-01	3.07e-01	6.54e-02	3.36e-01
	100	9.36e-01	7.75e-01	8.09e-01	9.47e-01	
	Total	6.97e-01	4.84e-01	5.05e-01	5.35e-01	
	10	**3.49e-11**	1.61e-01	3.34e-01	**2.86e-10**	
	25	**3.01e-06**	1.33e-02	8.97e-01	**2.75e-03**	
f21	50	7.85e-01	2.33e-01	5.62e-01	3.50e-01	
	75	7.87e-01	1.00e+00	6.34e-02	2.17e-01	2.28e-01
	100	8.77e-01	8.24e-01	1.07e-01	2.34e-01	
	Total	8.97e-02	3.21e-01	6.68e-01	6.21e-01	
	10	6.25e-10	1.61e-01	NA	**9.10e-12**	
	25	**2.49e-13**	**1.66e-11**	**1.95e-09**	**3.64e-11**	
f22	50	**3.78e-05**	**3.49e-09**	**2.56e-07**	**2.45e-06**	
	75	**1.92e-05**	4.67e-01	2.07e-02	7.64e-01	**1.99e-09**
	100	5.72e-01	9.59e-01	1.55e-01	7.08e-01	
	Total	**1.71e-04**	**3.37e-03**	**1.26e-03**	**9.07e-05**	
	10	**1.53e-04**	4.32e-01	2.63e-02	**6.88e-04**	
	25	4.85e-01	2.30e-01	5.23e-01	**2.88e-03**	
f23	50	6.02e-01	3.98e-01	5.14e-02	6.54e-01	
	75	2.93e-01	4.06e-01	3.85e-02	7.08e-01	3.66e-01
	100	8.09e-01	2.60e-01	3.00e-01	1.82e-01	
	Total	6.12e-01	9.90e-01	4.74e-01	5.36e-01	
	10	**6.26e-05**	6.11e-02	**6.49e-04**	**3.04e-03**	
	25	**6.64e-08**	**1.99e-04**	**7.24e-10**	**2.84e-07**	
f24	50	3.45e-02	4.79e-02	8.50e-02	2.34e-02	
	75	8.78e-01	2.19e-01	1.97e-01	9.94e-01	1.05e-01
	100	7.41e-01	5.33e-01	4.23e-01	4.23e-01	
	Total	3.14e-01	6.14e-01	3.17e-01	3.31e-01	
	10	4.74e-03	NA	1.61e-01	2.11e-02	
	25	**5.22e-12**	1.21e-12	1.21e-12	8.06e-11	
f25	50	5.04e-01	9.65e-02	1.55e-01	2.25e-02	
	75	2.24e-01	4.15e-02	1.09e-01	2.60e-01	1.14e-02
	100	4.40e-01	8.89e-01	4.32e-01	3.07e-01	
	Total	1.74e-01	1.24e-01	1.45e-01	2.91e-01	
	10	**1.29e-03**	**2.60e-03**	2.19e-01	1.50e-01	
	25	**9.21e-04**	**1.34e-04**	**1.40e-05**	1.05e-02	
f26	50	4.96e-02	1.12e-01	4.67e-01	6.33e-01	
	75	**6.34e-03**	5.13e-01	3.21e-01	7.23e-02	2.59e-01
	100	1.73e-01	1.92e-01	8.43e-01	7.41e-01	
	Total	3.59e-01	5.15e-01	7.59e-01	6.34e-01	

Continue page suivante

TABLEAU 4 : Résultats du test de Wilcoxon (p-valeur) par fonction et par pourcentage

		ABC	GABC	MeABC	qABC	Total
f27	10	**1.16e-08**	**6.49e-03**	7.20e-01	**8.14e-07**	1.54e-01
	25	5.62e-01	6.76e-01	1.34e-01	5.04e-01	
	50	6.00e-01	**7.90e-03**	7.07e-01	1.71e-01	
	75	3.33e-03	3.68e-01	2.38e-01	9.53e-01	
	100	8.66e-01	3.21e-01	9.35e-02	2.60e-01	
	Total	2.76e-01	5.36e-01	5.29e-01	3.38e-01	
f28	10	**7.36e-10**	4.46e-02	3.01e-01	**2.22e-08**	6.43e-02
	25	**3.45e-04**	1.20e-02	3.21e-02	**5.86e-05**	
	50	5.66e-01	2.50e-01	1.97e-01	2.55e-01	
	75	2.54e-01	4.11e-01	**6.09e-03**	8.43e-01	
	100	6.08e-01	NA	3.34e-01	1.27e-02	
	Total	1.76e-01	4.36e-01	9.49e-01	1.85e-01	
	Total	2.23e-02	2.21e-01	5.44e-01	2.04e-02	**1.10e-03**

B Figures d'évolution de l'erreur médiane pour qABC et qABC-Morris

Cette annexe présente différentes figures qui illustrent l'évolution de la valeur médiane de l'erreur pour les algorithmes qABC et qABCMorris, pour les cas de test de 50% et 25% de dimensions actives. Les relevés sont réalisés à intervalles réguliers du nombre d'évaluations. Les figures présentées sont les plus représentatives parmi les 28 fonctions du benchmark CEC 2013.

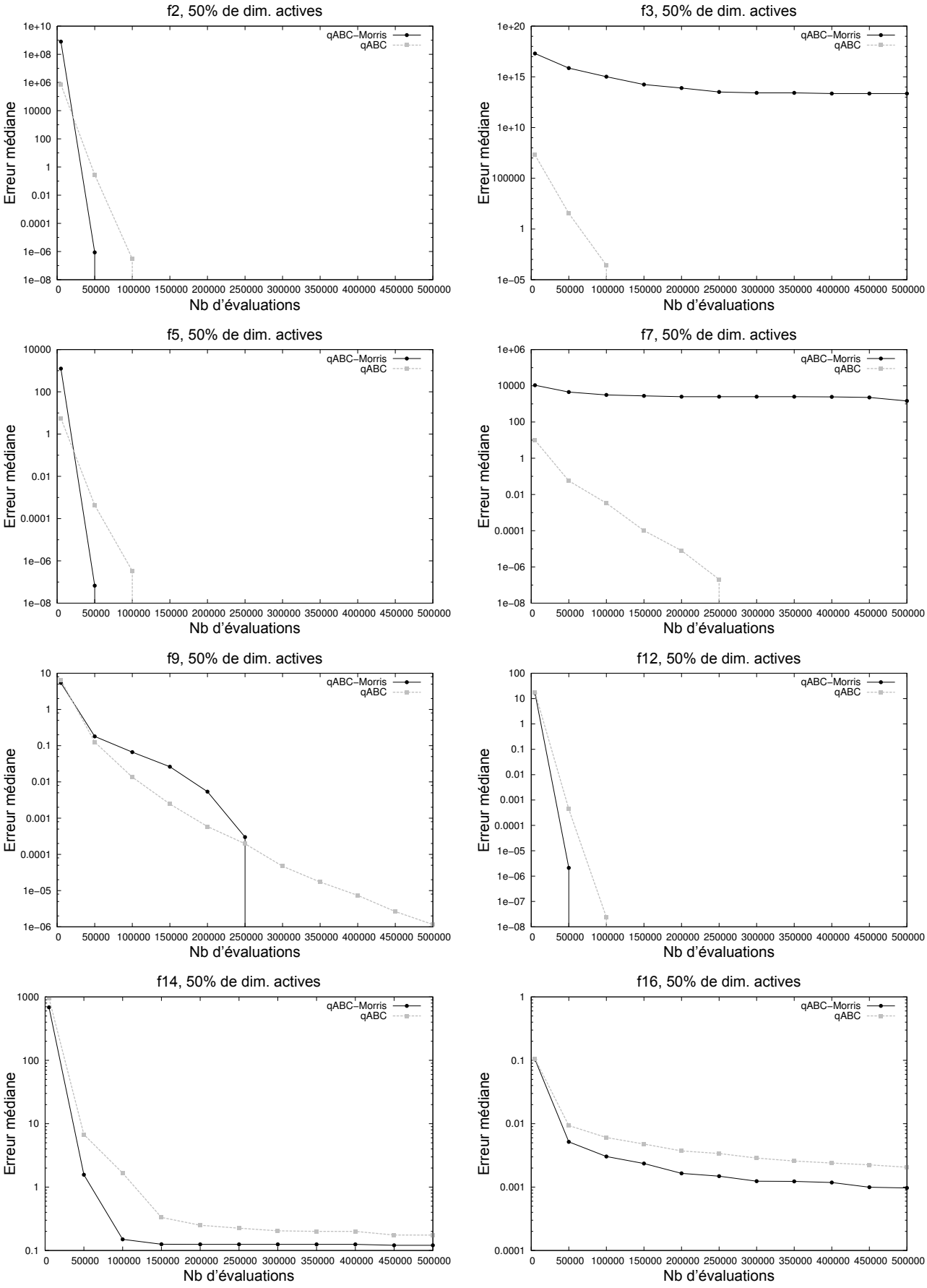

FIGURE 9 : Convergence pour qABC et qABCMorris - 50% de dimensions actives.

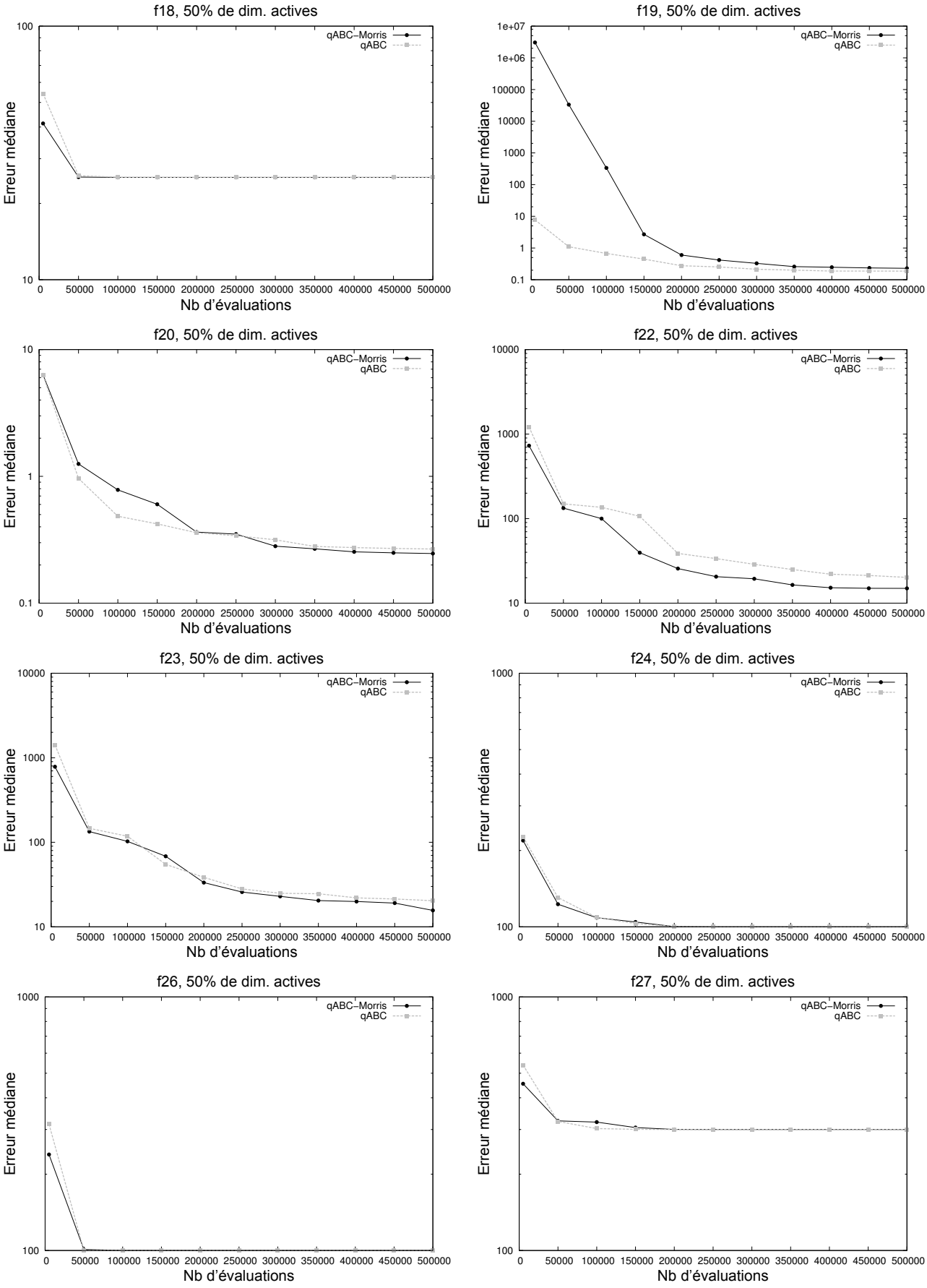

FIGURE 10 : Convergence pour qABC et qABCMorris - 50% de dimensions actives (suite).

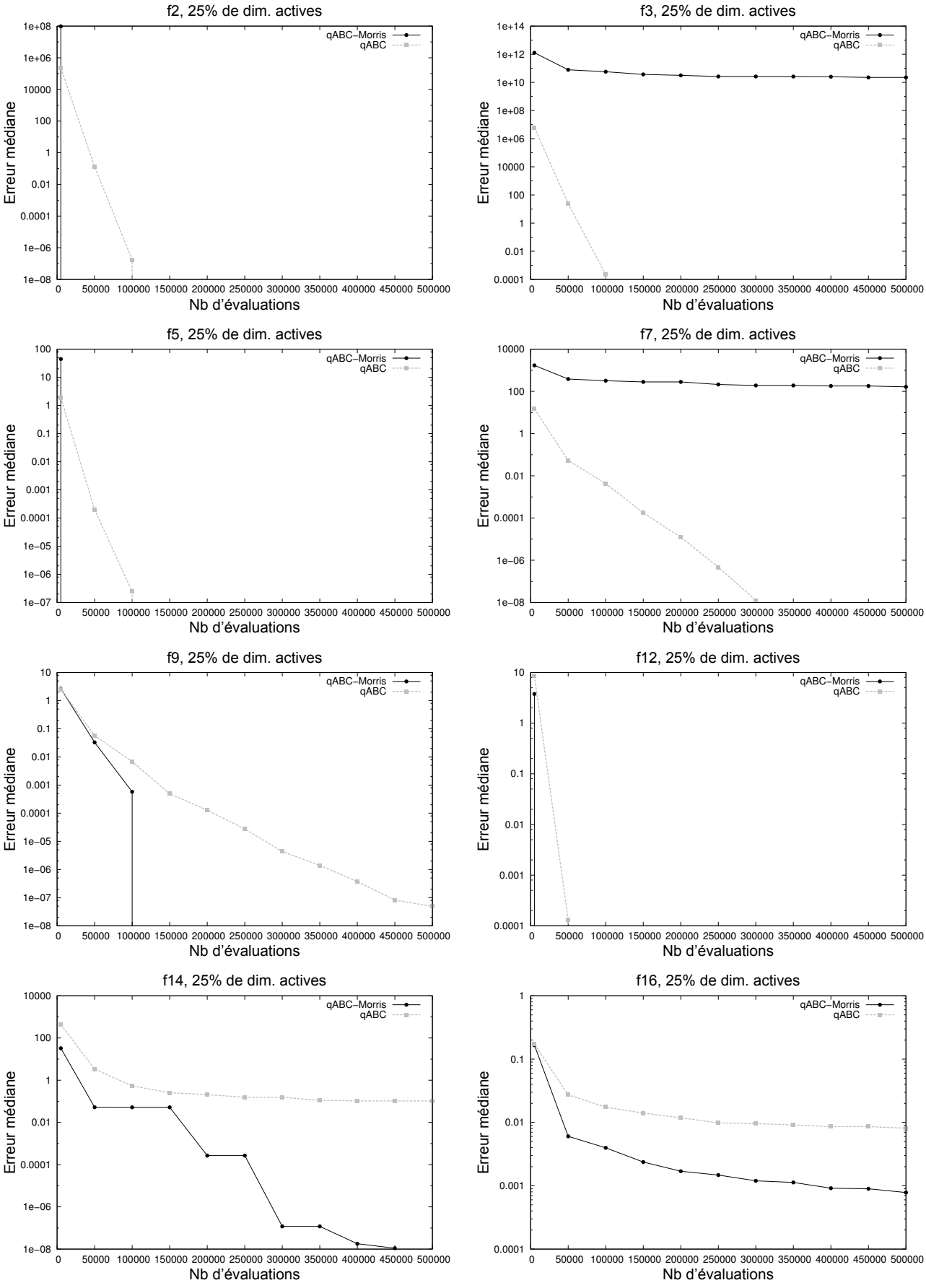

FIGURE 11 : Convergence pour qABC et qABCMorris - 25% de dimensions actives.

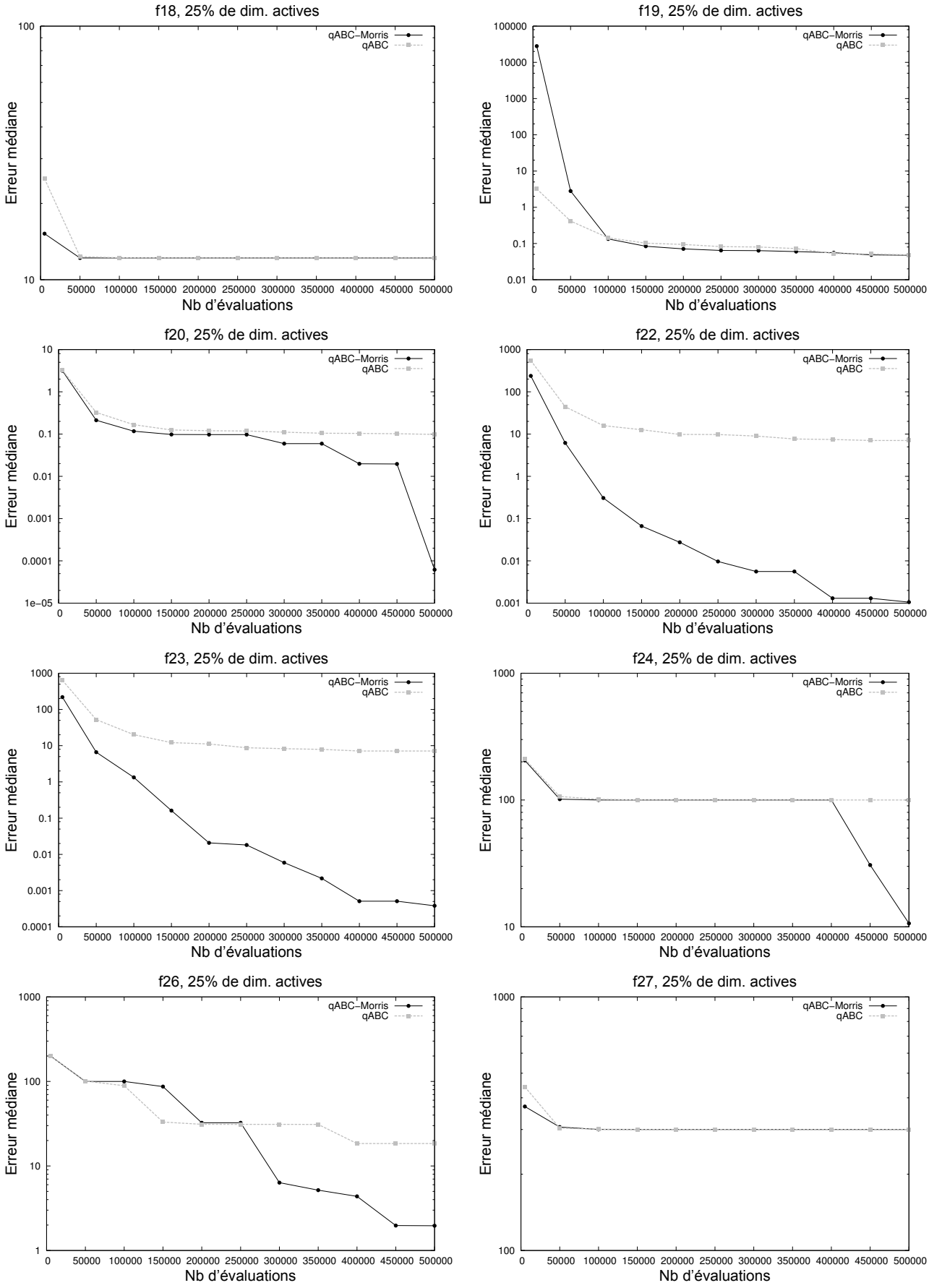

FIGURE 12 : Convergence pour qABC et qABCMorris - 25% de dimensions actives (suite).

C Scores des variables d'entrée pour NN-LCC et Morris

Cette annexe présente différentes figures qui illustrent l'évaluation des scores pour les variables d'entrée de plusieurs fonctions tests du domaine de l'analyse de sensibilité.

Chaque ligne est composée de deux figures, à gauche le résultat de l'évaluation des scores par la méthode NN-LCC et à droite, celui fournit par la méthode de Morris. Les quartiles sont calculés sur un ensemble de tests à paramètres différents. Les étoiles indiquent la valeur de poids déduite des indices de Sobol connus pour la fonction.

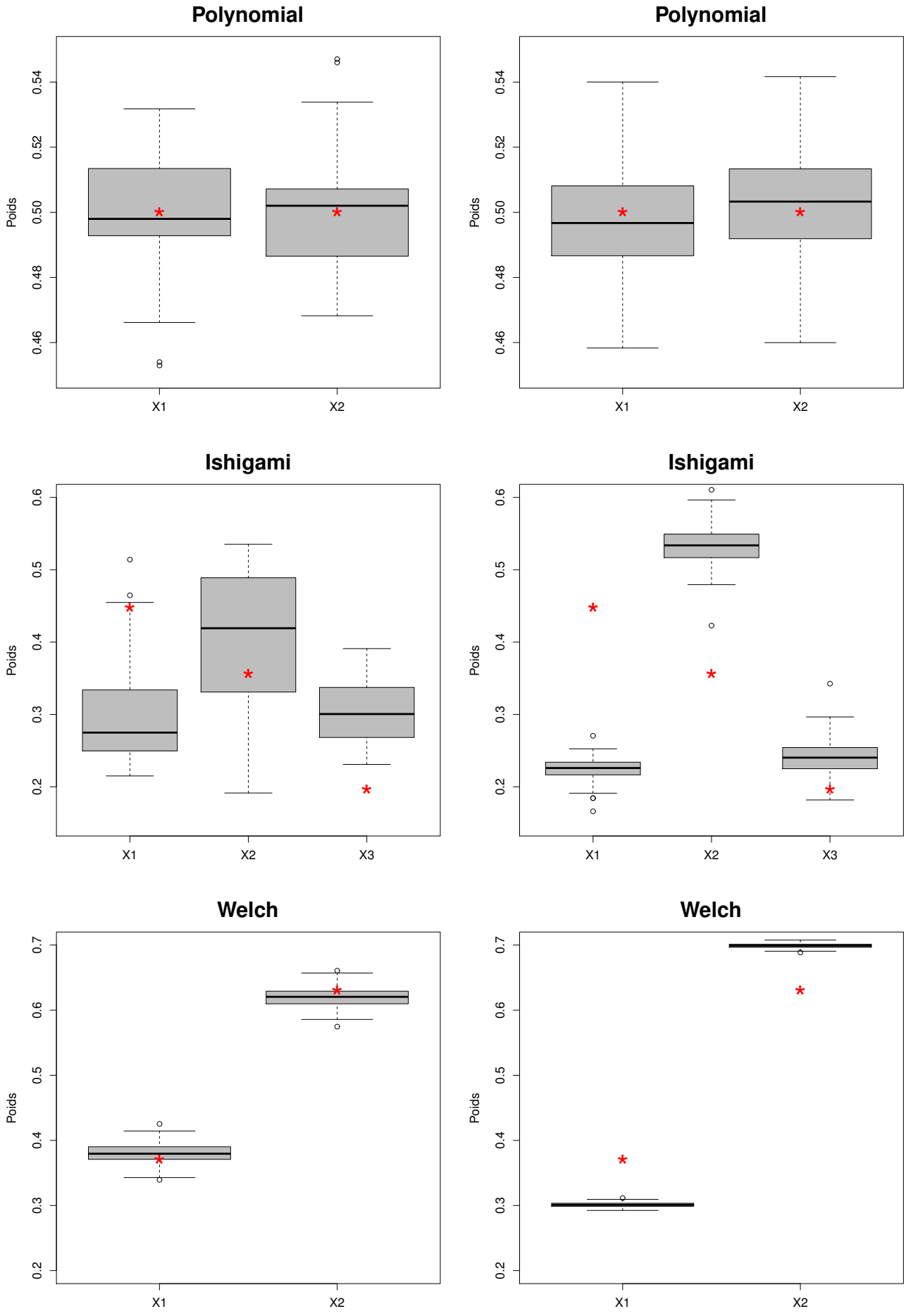

FIGURE 13 : Scores pour les variables d'entrée par les méthodes NN-LCC et Morris.

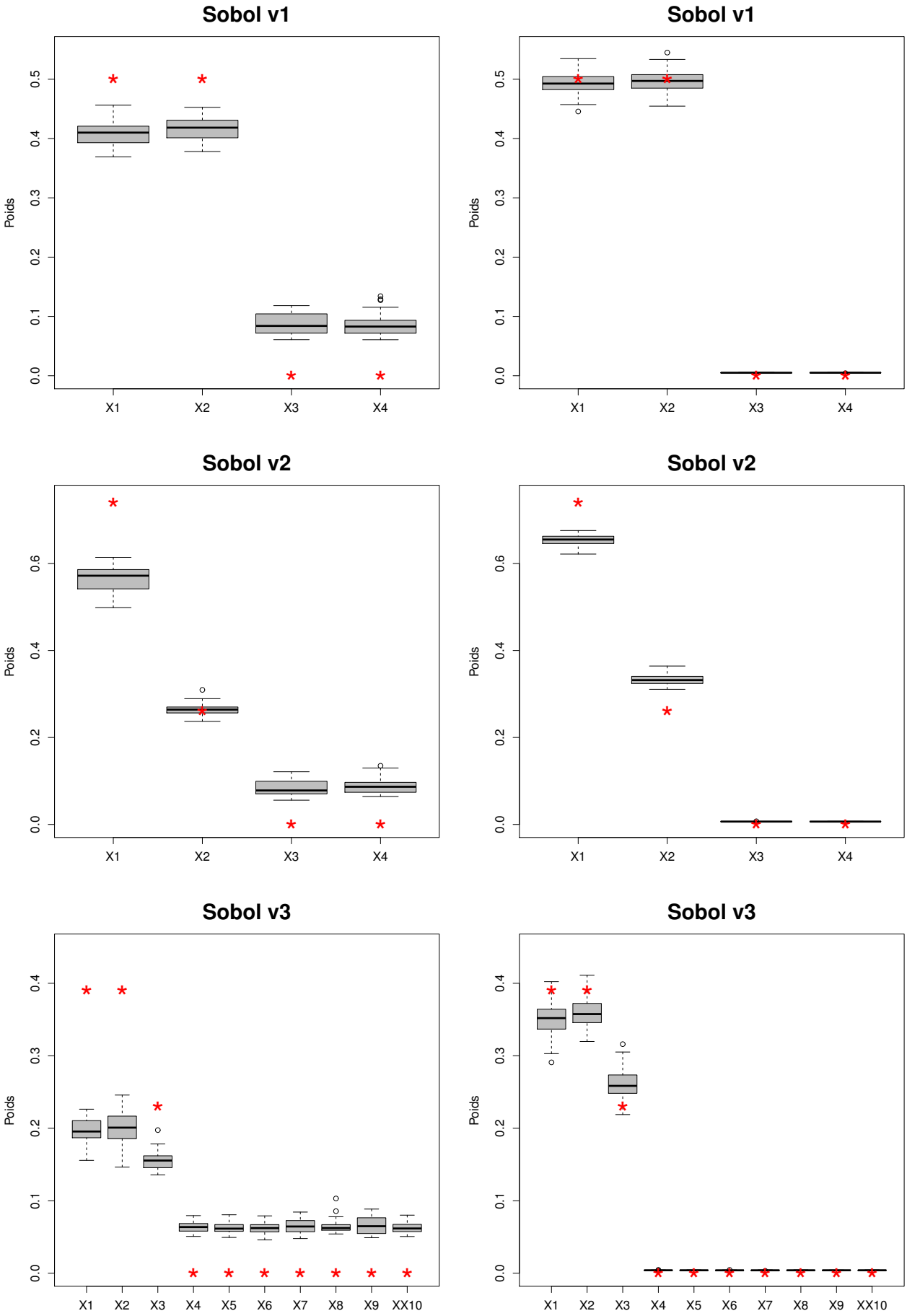

FIGURE 14 : Scores pour les variables d'entrée par les méthodes NN-LCC et Morris.

D Figures d'évolution de l'erreur médiane pour DE et DE-NN-LCC

Cette annexe présente différentes figures qui illustrent l'évolution de la valeur médiane de l'erreur pour les algorithmes DE et DE-NN-LCC, pour les cas de test de 25% de dimensions actives, en dimension 30 et 50. Les relevés sont réalisés à intervalles réguliers du nombre d'évaluations. Les figures présentées sont les plus représentatives parmi les 28 fonctions du benchmark CEC 2013.

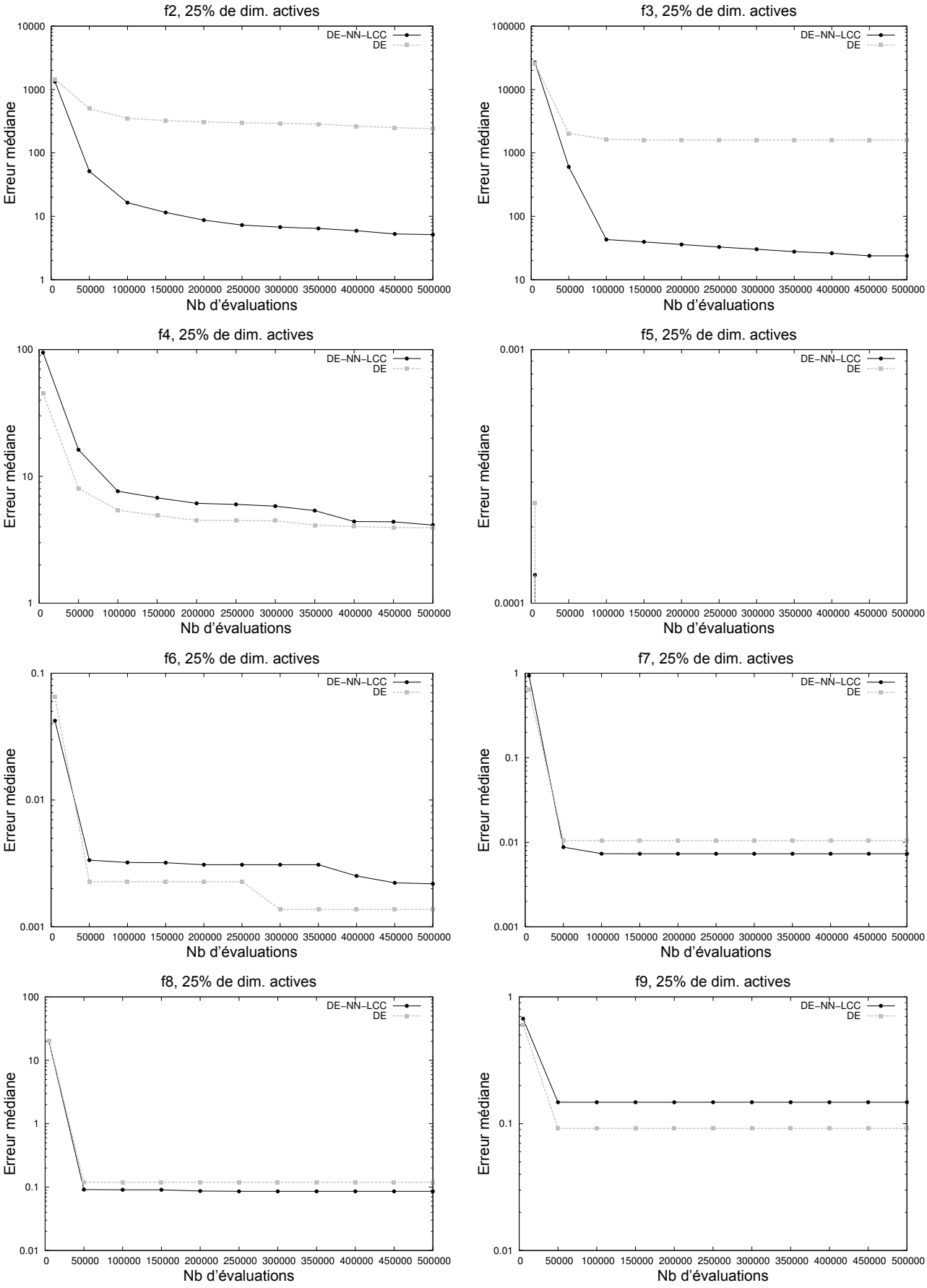

FIGURE 15 : Convergence pour DE et DE-NN-LCC - 25% de dimensions actives (D=30).

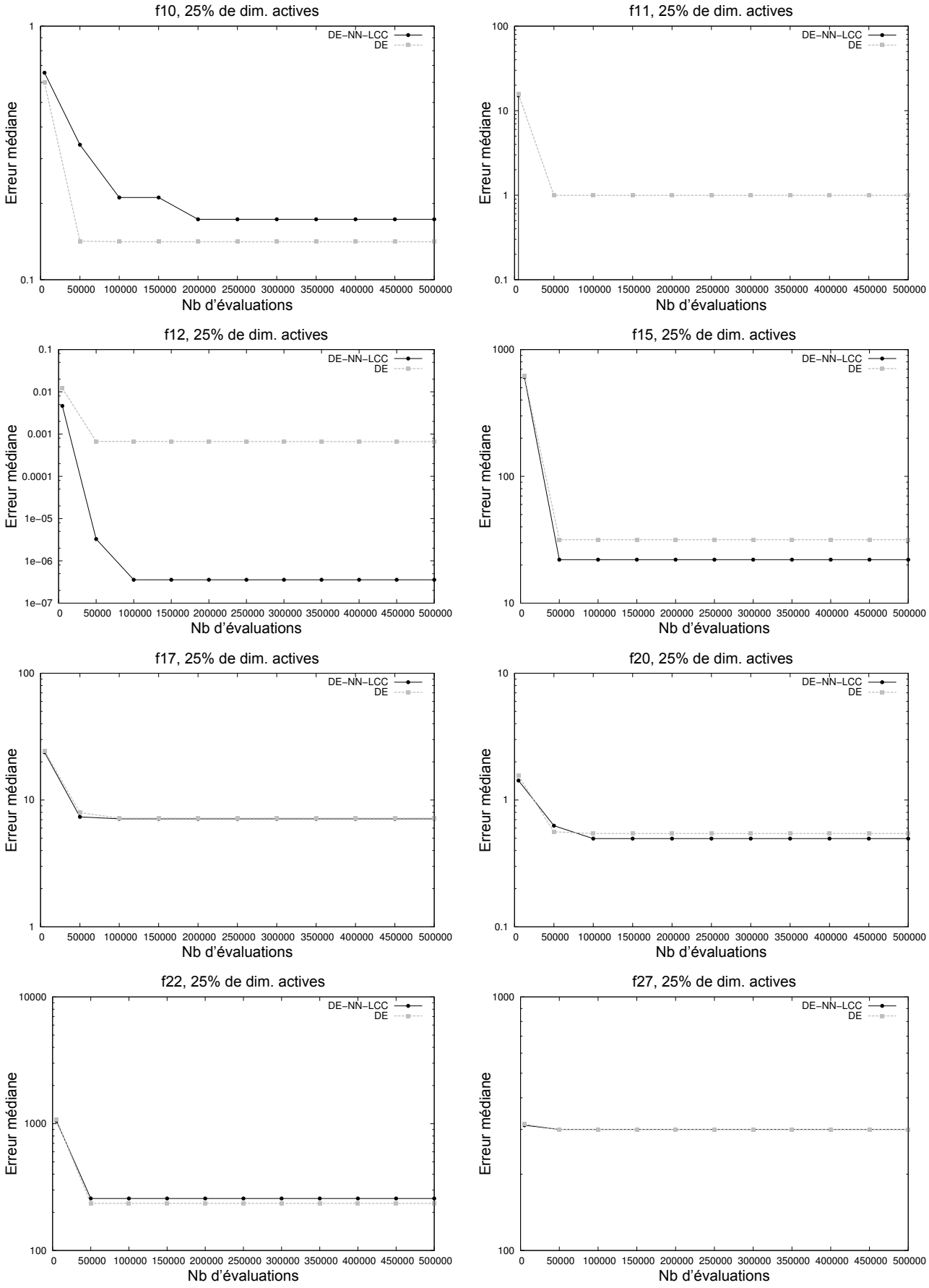

FIGURE 16 : Convergence pour DE et DE-NN-LCC - 25% de dimensions actives (D=30) (suite).

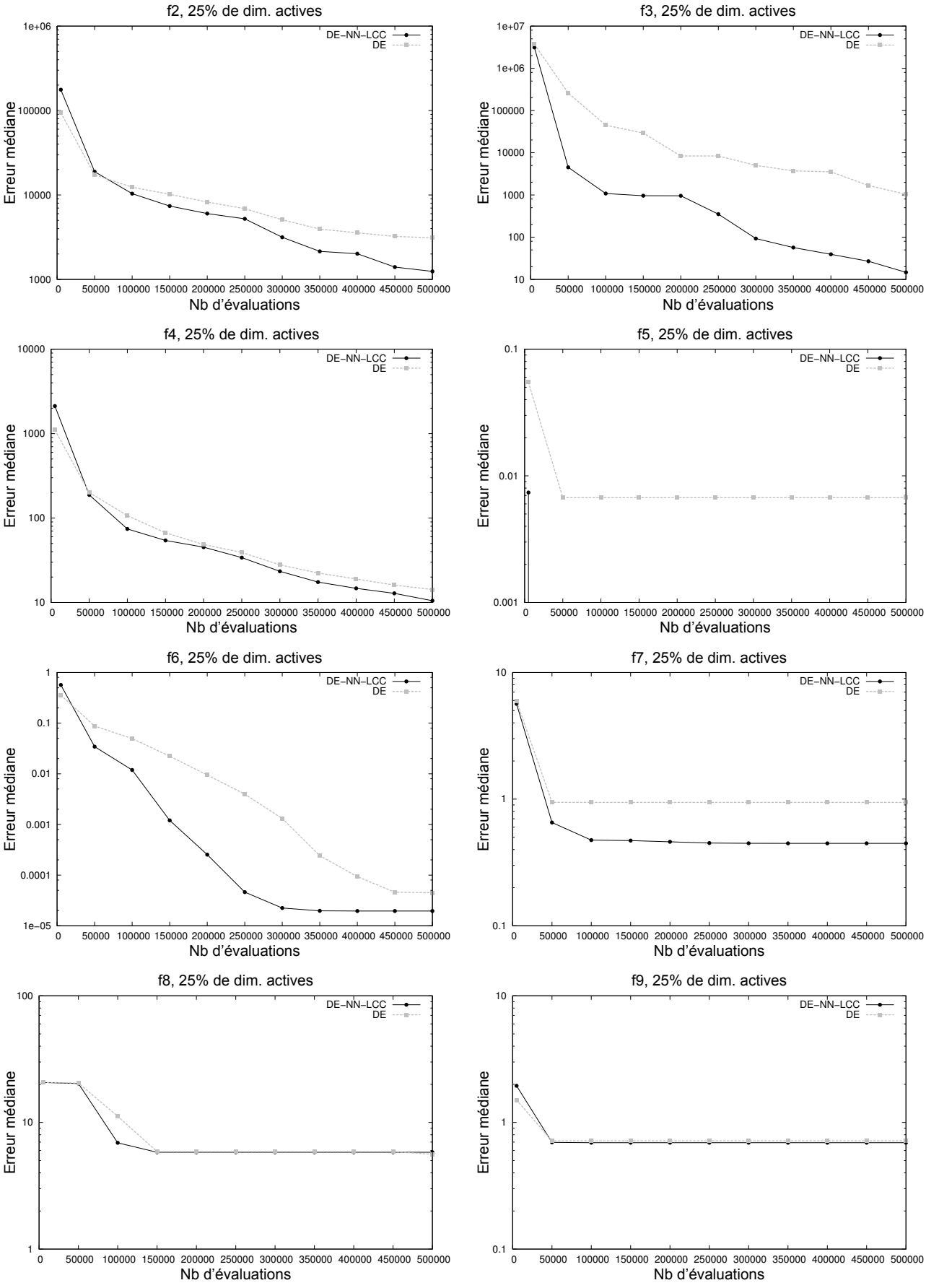

FIGURE 17 : Convergence pour DE et DE-NN-LCC - 25% de dimensions actives (D=50).

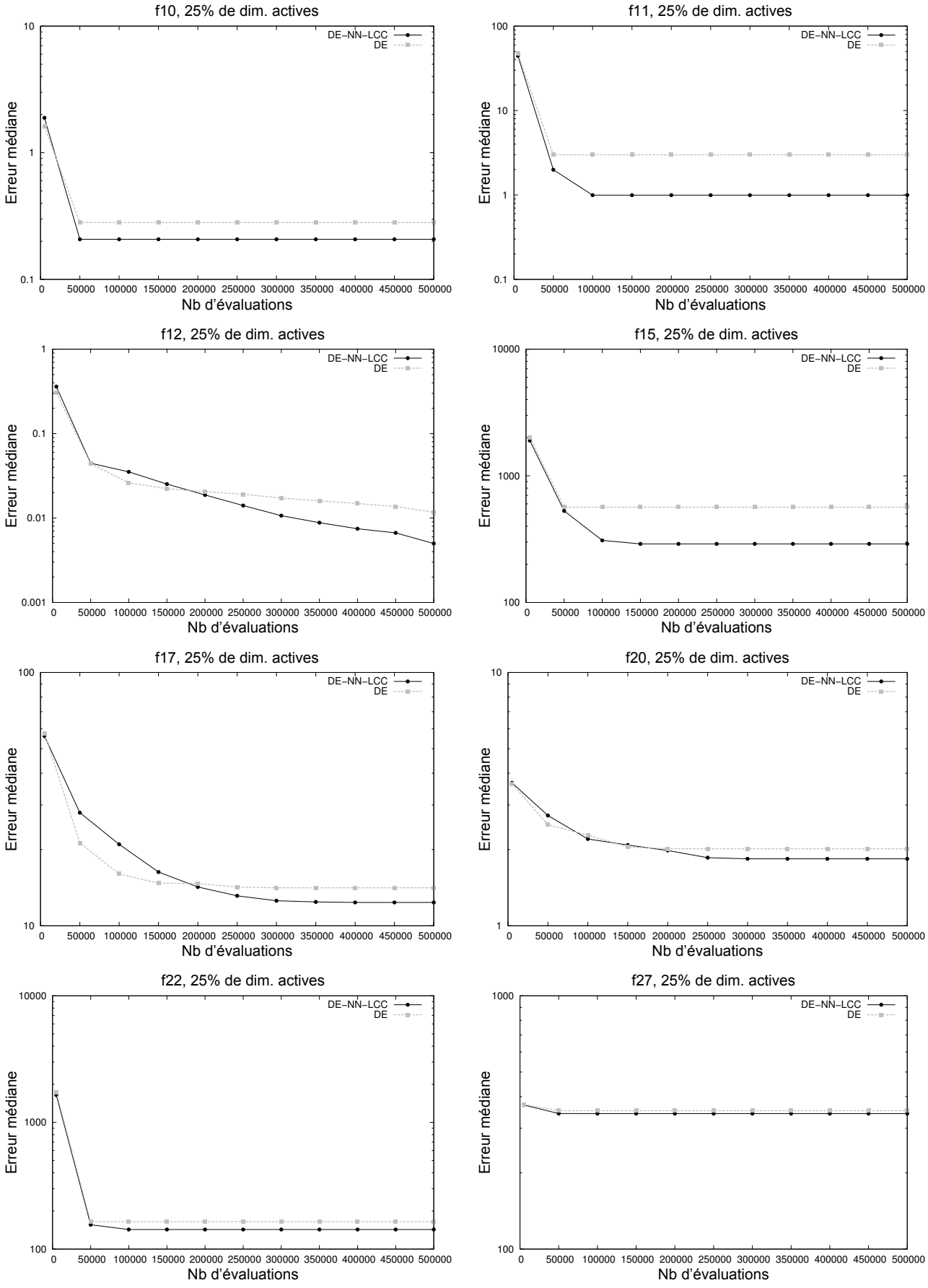

FIGURE 18 : Convergence pour DE et DE-NN-LCC - 25% de dimensions actives (D=50) (suite).

Références bibliographiques

Akay, B. et D. Karaboga. 2012, «A modified artificial bee colony algorithm for real-parameter optimization», *Information Sciences*, vol. 192, p. 120–142.

Avila, S., A. Lisboa, L. Krähenbühl, W. Carpes, J. Vasconcelos, R. Saldanha et R. Takahashi. 2006, «Sensitivity analysis applied to decision making in multiobjective evolutionary optimization», *IEEE Transactions on Magnetics*, vol. 42, n° 4, p. 1103–1106.

Baetens, R., B. P. Jelle et A. Gustavsen. 2010, «Phase change materials for building applications : A state-of-the-art review », *Energy and Buildings* , vol. 42, n° 9, p. 1361–1368.

Banharnsakun, A., T. Achalakul et B. Sirinaovakul. 2011, «The best-so-far selection in artificial bee colony algorithm», *Applied Soft Computing*, vol. 11, n° 2, p. 2888–2901.

Bansal, J. C., H. Sharma, K. V. Arya et A. Nagar. 2013, «Memetic search in Artificial Bee Colony algorithm», *Soft Computing*, vol. 17, n° 10, p. 1911–1928.

Bartz-Beielstein, T. et M. Preuss. 2007, «Experimental research in evolutionary computation», dans *Proceedings of the 9th Annual Conference Companion on Genetic and Evolutionary Computation, (chp 5), GECCO '07, Jul. 07–11 2007, London (UK)*, ACM, New York, NY, USA, ISBN 978-1-59593-698-1, p. 3001–3020.

Bast, H. et I. Weber. 2005, «Don't compare averages.», dans *WEA*, *Lecture Notes in Computer Science*, vol. 3503, édité par S. E. Nikoletseas, Springer, ISBN 3-540-25920-1, p. 67–76.

Bauer, L. et D. Hamby. 1991, «Relative sensitivities of existing and novel model parameters in atmospheric tritium dose estimates», *Radiation protection dosimetry*, vol. 37, n° 4, p. 253–260.

Bellman, R. E. 1957, *Dynamic Programming*, Princeton University Press, Rand Corporation, ISBN 978-0-691-07951-6.

Beni, G. et J. Wang. 1993, *Swarm Intelligence in Cellular Robotic Systems*, Springer, Berlin, Heidelberg, ISBN 978-3-642-58069-7, p. 703–712.

Van den Bergh, F. 2002, *An Analysis of Particle Swarm Optimizers*, thèse de doctorat, University of Pretoria, Pretoria, South Africa.

Van den Bergh, F. et A. P. Engelbrecht. 2004, «A cooperative approach to particle swarm optimization», *IEEE Transactions on Evolutionary Computation*, vol. 8, n° 3, p. 225–239.

Van den Bergh, F. et A. P. Engelbrecht. 2006, «A study of particle swarm optimization particle trajectories», *Information sciences*, vol. 176, n° 8, p. 937–971.

Bilchev, G. et I. C. Parmee. 1995, «The ant colony metaphor for searching continuous design spaces», dans *Selected Papers from AISB Workshop on Evolutionary Computing, Apr. 3–4 1995, Sheffield (UK)*, Springer, London, UK, UK, ISBN 3-540-60469-3, p. 25–39.

Birattari, M. et M. Dorigo. 2007, «How to assess and report the performance of a stochastic algorithm on a benchmark problem : mean or best result on a number of runs ?», *Optimization Letters*, vol. 1, n° 3, p. 309–311.

Bishop, C. M. 1995, *Neural networks for pattern recognition*, Oxford university press.

Blackwell, T. 2012, «A study of collapse in bare bones particle swarm optimization», *IEEE Transactions on Evolutionary Computation*, vol. 16, n° 3, p. 354–372.

Bonabeau, E., M. Dorigo et G. Theraulaz. 1999, «L'intelligence en essaim», dans *Ingénierie des systèmes Multi-Agents - JFIADSMA 99 - septième journées francophones d'Intelligence Artificielle et systèmes multi-agents, Nov. 8-10 1999, Le Récif, Saint Gilles, Ile de la Réunion (France)*, p. 25–38.

Box, G. E. P. 1952, «Multi-factor designs of first order», *Biometrika*, vol. 39, n° 1-2, p. 49–57.

Brest, J., S. Greiner, B. Boskovic, M. Mernik et V. Zumer. 2006, «Self-Adapting Control Parameters in Differential Evolution : A Comparative Study on Numerical Benchmark Problems», *IEEE Transactions on Evolutionary Computation*, vol. 10, n° 6, p. 646–657.

Campolongo, F., J. Cariboni et A. Saltelli. 2007, «An effective screening design for sensitivity analysis of large models», *Environmental Modelling & Software*, vol. 22, n° 10, p. 1509–1518.

Campos, M., R. A. Krohling et I. Enriquez. 2014, «Bare bones particle swarm optimization with scale matrix adaptation», *IEEE Transactions on Cybernetics*, vol. 44, n° 9, p. 1567–1578.

Charrier, R. 2009, *L'intelligence en essaim sous l'angle des systèmes complexes : étude d'un système multi-agent réactif à base d'itérations logistiques couplées*, thèse de doctorat, Université Nancy 2.

Chatterjee, A. et P. Siarry. 2006, «Nonlinear inertia weight variation for dynamic adaptation in particle swarm optimization», *Computers & Operations Research*, vol. 33, n° 3, p. 859–871.

Chelouah, R. et P. Siarry. 2000a, «A continuous genetic algorithm designed for the global optimization of multimodal functions», *Journal of Heuristics*, vol. 6, n° 2, p. 191–213.

Chelouah, R. et P. Siarry. 2000b, «Tabu search applied to global optimization», *European Journal of Operational Research*, vol. 123, n° 2, p. 256–270.

Chen, S.-M., A. Sarosh et Y.-F. Dong. 2012, «Simulated annealing based Artificial Bee Colony algorithm for global numerical optimization», *Applied Mathematics and Computation*, vol. 219, n° 8, p. 3575–3589.

Cheng, J., C. Yu et A. Zielen. 1991, «Resrad parameter sensitivity analysis», cahier de recherche, Argonne National Lab., IL (United States). Environmental Assessment and Information Sciences Div.

Clerc, M. 2003, «TRIBES - Un exemple d'optimisation par essaim particulaire sans paramètre de contrôle», dans *Conférence OEP'03, 2 Octobre 2003, Paris (France)*.

Clerc, M. 2006a, «Confinements and biases in particle swarm optimisation», *Particle Swarm Optimization, http://clerc.maurice.free.fr/pso*.

Clerc, M. 2006b, *Particle swarm optimization*, ISTE, London, Newport Beach, ISBN 1-905209-04-5.

Clerc, M. et J. Kennedy. 2002, «The particle swarm - explosion, stability, and convergence in a multidimensional complex space», *Transactions on Evolutionary Computation*, vol. 6, n° 1, p. 58–73.

Colorni, A., M. Dorigo, V. Maniezzo et collab.. 1992, «An investigation of some properties of an "ant algorithm".», dans *Proceedings of Parallel Problem Solving from Nature, PPSN'2, Sept. 28–30 1992, Brussels (Belgium)*, vol. 92, p. 509–520.

Commissariat général au développement durable. 2016, «Chiffres clés de l'énergie», http://www.developpement-durable.gouv.fr/IMG/pdf/reperes-chiffres-cles-energie-2015.pdf. Février 2016.

Cooren, Y. 2008, *Perfectionnement d'un algorithme adaptatif d'Optimisation par Essaim Particulaire. Application en génie médical et en électronique*, thèse de doctorat, Université Paris-Est-Créteil.

Cukier, R., H. Levine et K. Shuler. 1978, «Nonlinear sensitivity analysis of multiparameter model systems», *Journal of computational physics*, vol. 26, n° 1, p. 1–42.

Das, S., A. Abraham, U. K. Chakraborty et A. Konar. 2009, «Differential evolution using a neighborhood-based mutation operator», *IEEE Transactions on Evolutionary Computation*, vol. 13, n° 3, p. 526–553.

Das, S. et P. N. Suganthan. 2011, «Differential evolution : A survey of the state-of-the-art», *IEEE Transactions on Evolutionary Computation*, vol. 15, n° 1, p. 4–31.

Davis, L., éd.. 1991, *Handbook of Genetic Algorithms*, Van Nostrand Reinhold.

Derrac, J., S. García, D. Molina et F. Herrera. 2011, «A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms», *Swarm and Evolutionary Computation*, vol. 1, n° 1, p. 3–18.

Dorigo, M. et L. M. Gambardella. 1997, «Ant colony system : a cooperative learning approach to the traveling salesman problem», *IEEE Transactions on Evolutionary Computation*, vol. 1, n° 1, p. 53–66.

Dorigo, M. et T. Stützle. 2003, *The Ant Colony Optimization Metaheuristic : Algorithms, Applications, and Advances*, Springer, Boston, MA, ISBN 978-0-306-48056-0, p. 250–285.

Downing, D. J., R. Gardner et F. Hoffman. 1985, «An examination of response-surface methodologies for uncertainty analysis in assessment models», *Technometrics*, vol. 27, n° 2, p. 151–163.

Dréo, J. et P. Siarry. 2002, «A new ant colony algorithm using the heterarchical concept aimed at optimization of multiminima continuous functions», dans *Proceedings of the Third International Workshop on Ant Algorithms, ANTS'02, Sept. 12–14 2002, Brussels (Belgium)*, ANTS'02, Springer, London, UK, UK, ISBN 3-540-44146-8, p. 216–221.

Dréo, J. et P. Siarry. 2007, «Hybrid Continuous Interacting Ant Colony aimed at enhanced Global Optimization», *Algorithmic Operations Research*, vol. 2, n° 1, p. 52–64.

Dutil, Y., D. Rousse, S. Lassue, L. Zalewski, A. Joulin, J. Virgone, F. Kuznik, K. Johannes, J.-P. Dumas, J.-P. Bédécarrats et collab.. 2014, «Modeling phase change materials behavior in building applications : Comments on material characterization and model validation», *Renewable Energy*, vol. 61, p. 132–135.

Eberhart, R. et J. Kennedy. 1995, «A new optimizer using particle swarm theory», dans *Proceedings of the Sixth International Symposium on Micro Machine and Human Science, MHS '95, Oct. 4–6 1995, Nagoya (Japan)*, p. 39–43.

Eberhart, R. et Y. Shi. 2000, «Comparing inertia weights and contriction factors in particle swarm optimization», dans *Proceedings of the 6th IEEE Congress on Evolutionary Computation, CEC 2000, Jul. 16–19 2000, La Jolla (USA)*, vol. 1, IEEE Press, p. 84–88.

Eberhart, R., P. Simpson et R. Dobbins. 1996, *Computational Intelligence PC Tools*, Academic Press Professional, Inc., San Diego, CA, USA, ISBN 0-12-228630-8, 212–226 p..

El Dor, A. 2012, *Perfectionnement des algorithmes d'Optimisation par Essaim Particulaire. Application en segmentation d'images et en électronique*, thèse de doctorat, Université Paris-Est-Créteil.

El Dor, A., M. Clerc et P. Siarry. 2012, «Hybridization of differential evolution and particle swarm optimization in a new algorithm : DEPSO-2S», dans *Proceedings of the Swarm and Evolutionary Computation - International Symposia, SIDE 2012 and EC 2012, Held in Conjunction with ICAISC 2012, April 29–May 3 2012, Zakopane (Poland)*, p. 57–65.

El Dor, A. et P. Siarry. 2015, *Effect of the Dynamic Topology on the Performance of PSO-2S Algorithm for Continuous Optimization*, Springer, Cham, ISBN 978-3-319-27926-8, p. 60–64.

Engelbrecht, A. P. 2010, «Heterogeneous particle swarm optimization», dans *Proceedings of the 7th International Conference on Swarm Intelligence, ANTS 2010, Sept. 8–10 2010, Brussels (Belgium)*, Springer, ISBN 3-642-15460-3, p. 191–202.

Faivre, R., B. Iooss, S. Mahévas, D. Makowski et H. Monod. 2013, *Analyse de sensibilité et exploration de modèles*, Collection Savoir-Faire, Editions Quae.

Fang, K., R. Li et A. Sudjianto. 2006, *Design and modeling for computer experiments*, Computer science and data analysis series, Chapman & Hall/CRC, ISBN 1-584-88546-7.

Fister, I., I. F. Jr., J. Brest et V. Zumer. 2012, «Memetic Artificial Bee Colony algorithm for large-scale global optimization», dans *Proceedings of the IEEE Congress on Evolutionary Computation, CEC2012, Jun. 10–15 2012, Brisbane (Australia)*, IEEE, ISBN 978-1-4673-1510-4, p. 1–8.

Fogel, L. J., A. J. Owens et M. J. Walsh. 1966, *Artificial Intelligence through Simulated Evolution*, John Wiley.

Franquet, E., S. Gibout, J.-P. Bédécarrats, D. Haillot et J.-P. Dumas. 2012, «Inverse method for the identification of the enthalpy of phase change materials from calorimetry experiments», *Thermochimica acta*, vol. 546, p. 61–80.

Gämperle, R., S. D. Müller et P. Koumoutsakos. 2002, «A parameter study for differential evolution», *Advances in Intelligent Systems, Fuzzy Systems, Evolutionary Computation*, vol. 10, p. 293–298.

Gardeux, V., R. Chelouah, P. Siarry et F. Glover. 2009, «Unidimensional search for solving continuous high-dimensional optimization problems», dans *Proceedings of the 9th International Conference on Intelligent Systems Design and Applications, ISDA'09, Nov. 30–Dec. 2 2009, Pisa (Italy)*, IEEE, p. 1096–1101.

Gibout, S., E. Franquet, W. Maréchal et J.-P. Dumas. 2013, «On the use of a reduced model for the simulation of melting of solutions in DSC experiments», *Thermochimica Acta*, vol. 566, p. 118–123.

Glover, F. 1986, «Applications of integer programming future paths for integer programming and links to artificial intelligence», *Computers & Operations Research*, vol. 13, n° 5, p. 533 – 549.

Goldberg, D. E. et J. H. Holland. 1988, «Genetic algorithms and machine learning», *Machine learning*, vol. 3, n° 2, p. 95–99.

Goss, S., S. Aron, J. L. Deneubourg et J. M. Pasteels. 1989, «Self-organized shortcuts in the Argentine ant», *Naturwissenschaften*, vol. 76, p. 579–581.

Gosselin, L., M. Tye-Gingras et F. Mathieu-Potvin. 2009, «Review of utilization of genetic algorithms in heat transfer problems», *International Journal of Heat and Mass Transfer*, vol. 52, n° 9, p. 2169–2188.

Hansen, N. et A. Ostermeier. 2001, «Completely derandomized self-adaptation in evolution strategies», *Evolutionary computation*, vol. 9, n° 2, p. 159–195.

Hastings, W. K. 1970, «Monte-Carlo sampling methods using Markov chains and their applications», *Biometrika*, vol. 57, n° 1, p. 97–109.

Holland, J. H. 1975, *Adaptation in natural and artificial systems : an introductory analysis with applications to biology, control, and artificial intelligence.*, U Michigan Press.

Imanian, N., M. E. Shiri et P. Moradi. 2014, «Velocity based Artificial Bee Colony algorithm for high dimensional continuous optimization problems», *Engineering Applications of Artificial Intelligence*, vol. 36, n° 0, p. 148–163.

Iooss, B. 2011, «Revue sur l'analyse de sensibilité globale de modèles numériques», *Journal de la Société Française de Statistique*, vol. 152, n° 1, p. 3–25.

Iooss, B., L. Boussouf, V. Feuillard et A. Marrel. 2010, «Numerical studies of the metamodel fitting and validation processes», *arXiv preprint arXiv :1001.1049*.

Iooss, B. et P. Lemaître. 2015, «A review on global sensitivity analysis methods», dans *Uncertainty Management in Simulation-Optimization of Complex Systems : Algorithms and Applications*, édité par G. Dellino et C. Meloni, Springer, Boston, MA, ISBN 978-1-4899-7547-8, p. 101–122.

Jadon, S. S. 2015, *Enhanced Artificial Bee Colony Algorithms and their Applications*, thèse de doctorat, ABV Indian Institute of Information Technology and Management Gwalior, MP, India.

Jadon, S. S., J. C. Bansal, R. Tiwari et H. Sharma. 2014, «Artificial bee colony algorithm with global and local neighborhoods», *International Journal of System Assurance Engineering and Management*, p. 1–13.

Jansen, M. J. 1999, «Analysis of variance designs for model output», *Computer Physics Communications*, vol. 117, n° 1, p. 35–43.

Jourdan, A. 2012, «Global sensitivity analysis using complex linear models», *Statistics and Computing*, vol. 22, n° 3, p. 823–831.

Jourdan, A. et J. Franco. 2010, «Optimal Latin hypercube designs for the Kullback-Leibler criterion», *AStA Advances in Statistical Analysis*, vol. 94, n° 4, p. 341–351.

Karaboga, D. 2005, «An idea based on honey bee swarm for numerical optimization», cahier de recherche TR06, Erciyes University.

Karaboga, D. et B. Akay. 2009, «A comparative study of Artificial Bee Colony algorithm», *Applied Mathematics and Computation*, vol. 214, n° 1, p. 108–132.

Karaboga, D. et B. Basturk. 2008, «On the Performance of Artificial Bee Colony (ABC) algorithm», *Applied Soft Computing*, vol. 8, n° 1, p. 687–697.

Karaboga, D. et B. Gorkemli. 2014, «A quick Artificial Bee Colony (qABC) algorithm and its performance on optimization problems», *Applied Soft Computing*, vol. 23, n° 0, p. 227–238.

Karaboga, D., B. Gorkemli, C. Ozturk et N. Karaboga. 2012, «A comprehensive survey : Artificial Bee Colony (ABC) algorithm and applications», *Artificial Intelligence Review*, vol. 42, n° 1, p. 21–57.

Kaveh, A. 2014, «Particle swarm optimization», dans *Advances in Metaheuristic Algorithms for Optimal Design of Structures*, Springer, p. 9–40.

Kennedy, J. 1999, «Small worlds and mega-minds : effects of neighborhood topology on particle swarm performance», dans *Proceedings of the 1999 Congress on Evolutionary Computation, CEC 99, Jul. 6–9 1999, Washington DC (USA)*, vol. 3, p. 1931–1938.

Kennedy, J. 2003, «Bare bones particle swarms», dans *Proceedings of the 2003 IEEE Swarm Intelligence Symposium, Sis '03, Apr. 24–26 2003, Indianapolis (USA)*, p. 80–87.

Kennedy, J. et R. Eberhart. 1995, «Particle swarm optimization», dans *Proceedings of the IEEE International Conference on Neural Networks, Nov. 27–Dec. 1 1995, Perth, WA (Australia)*, vol. 4, IEEE, ISBN 0-7803-2768-3, p. 1942–1948.

Kennedy, J., R. C. Eberhart et Y. Shi. 2001, *Swarm Intelligence*, The Morgan Kaufmann Series in Artificial Intelligence, Morgan Kaufmann, San Francisco (USA), ISBN 978-1-55860-595-4.

Kennedy, J. et R. Mendes. 2002, «Population structure and particle swarm performance», dans *Proceedings of the 2002 Congress on Evolutionary Computation, CEC 2002, May 12–17 2002, Honolulu (Hawaii)*, vol. 2, p. 1671–1676.

Keogh, E. et A. Mueen. 2010, *Curse of Dimensionality*, Springer, Boston, MA, ISBN 978-0-387-30164-8, p. 257–258.

Kiefer, J. 1953, «Sequential minimax search for a maximum», *Proceedings of the American mathematical society*, vol. 4, n° 3, p. 502–506.

Kiran, M. S. et O. Findik. 2015, «A directed artificial bee colony algorithm», *Applied Soft Computing*, vol. 26, p. 454–462.

Kiran, M. S. et M. Gündüz. 2012, «A novel artificial bee colony-based algorithm for solving the numerical optimization problems», *International Journal of Innovative Computing, Information and Control*, vol. 8, n° 9, p. 6107–6121.

Kiran, M. S., H. Hakli, M. M. Gündüz et H. Uguz. 2015, «Artificial bee colony algorithm with variable search strategy for continuous optimization», *Information Sciences*, vol. 300, p. 140–157.

Kirkpatrick, S., C. D. Gelatt et M. P. Vecchi. 1983, «Optimization by simulated annealing», *Science*, vol. 220, n° 4598, p. 671–680.

Koza, J. R. 1989, «Hierarchical genetic algorithms operating on populations of computer programs.», dans *Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI' 11, Aug. 20–26 1999, Detroit (USA)*, édité par M. Kaufmann, p. 768–774.

Koza, J. R. 1990, «Genetic programming : A paradigm for genetically breeding populations of computer programs to solve problems», cahier de recherche STANCS-90-1314, Stanford University, Department of Computer Science.

Krohling, R. A. et E. Mendel. 2009, «Bare Bones Particle Swarm Optimization with Gaussian or Cauchy jumps», dans *Proceedings of the 2009 IEEE Congress on Evolutionary Computation, CEC 2009, May 18–21 2009, Trondheim (Norway)*, p. 3285–3291.

Kumar, S., V. K. Sharma et R. Kumari. 2013, «A novel hybrid crossover based artificial bee colony algorithm for optimization problem», *International Journal of Computer Applications*, vol. 82, n° 8, p. 18–25.

Lampinen, J. 2001, «A bibliography of differential evolution algorithm», *Lappeenranta University of Technology, Finland*.

Lane, J., A. Engelbrecht et J. Gain. 2008, «Particle swarm optimization with spatially meaningful neighbours», dans *Proceedings of the IEEE Swarm Intelligence Symposium, SIS 2008, Sep. 21–23 2008, St. Louis (USA)*, p. 1–8.

Leguizamón, G. et C. A. C. Coello. 2010, «An Alternative $\text{ACO}_{\mathbb{R}}$ Algorithm for Continuous Optimization Problems», dans *Proceedings of the International Conference on Swarm Intelligence, ICSI2010, Jun. 12–15 2010, Beijing (China)*, Springer, p. 48–59.

Lepagnot, J. 2011, *Conception de métaheuristiques pour l'optimisation dynamique. Application à l'analyse de séquences d'images IRM*, thèse de doctorat, Université Paris-Est.

Li, G., P. Niu et X. Xiao. 2012, «Development and investigation of efficient artificial bee colony algorithm for numerical function optimization», *Applied Soft Computing*, vol. 12, n° 1, p. 320–332.

Li, X., K. Tang, M. N. Omidvar, Z. Yang et K. Qin. 2013, «Benchmark functions for the CEC 2013 special session and competition on large-scale global optimization», .

Liang, J. J. et P. N. Suganthan. 2005, «Dynamic multi-swarm particle swarm optimizer», dans *Proceedings of the 2005 IEEE Swarm Intelligence Symposium, SIS 2005, Jun. 8–10 2005, Pasadena (USA)*, p. 124–129.

Liao, T., M. A. Montes de Oca, D. Aydin, T. Stützle et M. Dorigo. 2011, «An incremental ant colony algorithm with local search for continuous optimization», dans *Proceedings of the 13th annual conference on Genetic and evolutionary computation, GECCO'11, Jul. 12–16 2011, Dublin (Ireland)*, p. 125–132.

Liao, T., T. Stützle, M. A. M. de Oca et M. Dorigo. 2014, «A unified ant colony optimization algorithm for continuous optimization», *European Journal of Operational Research*, vol. 234, n° 3, p. 597–609.

Lim, W. H. et N. A. M. Isa. 2014, «Particle swarm optimization with increasing topology connectivity», *Engineering Applications of Artificial Intelligence*, vol. 27, p. 80–102.

Loubière, P., A. Jourdan, P. Siarry et R. Chelouah. 2016a, «A modified sensitivity analysis method for driving a multidimensional search in the artificial bee colony algorithm», dans *Proceedings of the IEEE Congress on Evolutionary Computation, CEC2016 (IEEE World Congress on Computational Intelligence), Jul. 24–29 2016, Vancouver (Canada)*.

Loubière, P., A. Jourdan, P. Siarry et R. Chelouah. 2016b, «A sensitivity analysis method aimed at enhancing the metaheuristics for continuous optimization», *Submitted to Artificial Intelligence Review, Avril 2016*.

Loubière, P., A. Jourdan, P. Siarry et R. Chelouah. 2016c, ≪A sensitivity analysis method for driving the artificial bee colony algorithm's search process≫, *Applied Soft Computing*, vol. 41, p. 515–531.

Lynn, N., R. Mallipeddi et P. N. Suganthan. 2015, *Differential Evolution with Two Subpopulations*, Springer, Cham, ISBN 978-3-319-20294-5, p. 1–13.

Mallipeddi, R., P. Suganthan, Q. Pan et M. Tasgetiren. 2011, ≪Differential evolution algorithm with ensemble of parameters and mutation strategies≫, *Applied Soft Computing*, vol. 11, n° 2, p. 1679–1696. The Impact of Soft Computing for the Progress of Artificial Intelligence.

Mandelbrot, B. B. 1977, *The Fractal Geometry of Nature*, W. H. Freeman and Company, 289–290 p..

Maréchal, W. Septembre 2014, *Utilisation de méthodes inverses pour la caractérisation de matériaux à changement de phase*, thèse de doctorat, Université de Pau et des Pays de l'Adour.

Mendes, R., J. Kennedy et J. Neves. 2004, ≪The fully informed particle swarm : simpler, maybe better≫, *IEEE Transactions on Evolutionary Computation*, vol. 8, n° 3, p. 204–210.

Mendes, R. et J. Neves. 2004, *What Makes a Successful Society ?*, Springer, Berlin, Heidelberg, ISBN 978-3-540-28645-5, p. 346–355.

Mernik, M., S. Liu, D. Karaboga et M. Crepinsek. 2015, ≪On clarifying misconceptions when comparing variants of the artificial bee colony algorithm by offering a new implementation≫, *Information Sciences*, vol. 291, p. 115–127.

Michalewicz, Z. 1996, *Genetic Algorithms + Data Structures = Evolution Programs (3rd Ed.)*, Springer, London, UK, UK, ISBN 3-540-60676-9.

Mohais, A. S., R. Mendes, C. Ward et C. Posthoff. 2005, *Neighborhood Re-structuring in Particle Swarm Optimization*, Springer, Berlin, Heidelberg, ISBN 978-3-540-31652-7, p. 776–785.

Monmarché, N., G. Venturini et M. Slimane. 2000, ≪On how pachycondyla apicalis ants suggest a new search algorithm≫, *Future Generation Computer Systems*, vol. 16, n° 8, p. 937–946.

Montgomery, D. C. 2008, *Design and analysis of experiments*, John Wiley & Sons.

Montgomery, J. 2010, ≪Crossover and the different faces of differential evolution searches≫, dans *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2010, Jul. 18–23 2010, Barcelona (Spain)*, p. 1–8.

Morris, M. D. 1991, ≪Factorial sampling plans for preliminary computational experiments≫, *Technometrics*, vol. 33, n° 2, p. 161–174.

Myers, R. H., D. C. Montgomery et C. M. Anderson-Cook. 2016, *Response surface methodology : process and product optimization using designed experiments*, John Wiley & Sons.

Nelder, J. A. et R. Mead. 1965, ≪A simplex method for function minimization≫, *The Computer Journal*, vol. 7, n° 4, p. 308–313.

Neri, F. et V. Tirronen. 2010, ≪Recent advances in differential evolution : A survey and experimental analysis≫, *Artificial Intelligence Review*, vol. 33, n° 1-2, p. 61–106.

Niu, B., Y. Zhu, X. He et H. Wu. 2007, ≪Mcpso : A multi-swarm cooperative particle swarm optimizer≫, *Applied Mathematics and Computation*, vol. 185, n° 2, p. 1050–1062. Special Issue on Intelligent Computing Theory and Methodology.

Nkwetta, D. N. et F. Haghighat. 2014, ≪Thermal energy storage with phase change material – a state-of-the art review≫, *Sustainable cities and society*, vol. 10, p. 87–100.

Norkin, V. I., G. C. Pflug et A. Ruszczyński. 1998, ≪A branch and bound method for stochastic global optimization≫, *Mathematical Programming*, vol. 83, n° 1, p. 425–450.

Montes de Oca, M. A. et T. Stützle. 2008, ≪Convergence Behavior of the Fully Informed Particle Swarm Optimization Algorithm≫, dans *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation, GECCO'08, Jul. 12–16 2008, Atlanta (USA)*, ISBN 978-1-60558-130-9, p. 71–78.

Montes de Oca, M. A., T. Stutzle, M. Birattari et M. Dorigo. 2009, ≪Frankenstein's PSO : A Composite Particle Swarm Optimization Algorithm≫, *IEEE Transactions on Evolutionary Computation*, vol. 13, n° 5, p. 1120–1132.

Ozkis, A. et A. Babalik. 2013, ≪Accelerated ABC (A-ABC) Algorithm for Continuous Optimization Problems≫, *LNSE : Lecture Notes on Software Engineering*, vol. 1, n° 3, p. 262–266.

Parsopoulos, K. et M. Vrahatis. 2007, ≪Parameter selection and adaptation in unified particle swarm optimization≫, *Mathematical and Computer Modelling*, vol. 46, n° 1-2, p. 198–213. Proceedings of the International Conference on Computational Methods in Sciences and Engineering, Jul. 24–28 2004, Jyväskylä (Finland).

Parsopoulos, K. E. et M. N. Vrahatis. 2005, *Unified Particle Swarm Optimization in Dynamic Environments*, Springer, Berlin, Heidelberg, ISBN 978-3-540-32003-6, p. 590–599.

Passino, K. M. 2002, «Biomimicry of bacterial foraging for distributed optimization and control», *IEEE control systems*, vol. 22, n° 3, p. 52–67.

Plischke, E., E. Borgonovo et C. L. Smith. 2013, «Global sensitivity measures from given data», *European Journal of Operational Research*, vol. 226, n° 3, p. 536–550.

Price, K., R. M. Storn et J. A. Lampinen. 2005, *Differential Evolution : A Practical Approach to Global Optimization (Natural Computing Series)*, Springer, ISBN 3540209506.

Price, K. V. 1999, «An introduction to differential evolution», dans *New Ideas in Optimization*, chap. An Introduction to Differential Evolution, McGraw-Hill Ltd., ISBN 0-07-709506-5, p. 79–108.

Qin, A. K., V. L. Huang et P. N. Suganthan. 2009, «Differential evolution algorithm with strategy adaptation for global numerical optimization», *IEEE Transactions on Evolutionary Computation*, vol. 13, n° 2, p. 398–417.

Qin, Q., S. Cheng, Q. Zhang, L. Li et Y. Shi. 2015, «Artificial bee colony algorithm with time-varying strategy», *Discrete Dynamics in Nature and Society*, vol. 2015.

Rahnamayan, S., H. R. Tizhoosh et M. M. A. Salama. 2008, *Opposition-Based Differential Evolution*, Springer, Berlin, Heidelberg, ISBN 978-3-540-68830-3, p. 155–171.

Ranganathan, A. 2004, «The Levenberg-Marquardt Algorithm», http://ananth.in/docs/lmtut.pdf. 8 juin 2004.

Rao, S. S. et S. Rao. 2009, *Engineering optimization : theory and practice*, John Wiley & Sons.

Rechenberg, I. 1989, *Evolution strategy : Nature's way of optimization*, Springer, 106–126 p..

S. Das, P. N. S., S. S. Mullick. 2016, «Recent advances in differential evolution - an updated survey», *Swarm and Evolutionary Computation*, vol. 27, p. 1–30.

Saltelli, A. 2002a, «Making best use of model evaluations to compute sensitivity indices», *Computer Physics Communications*, vol. 145, n° 2, p. 280–297.

Saltelli, A. 2002b, «Sensitivity analysis for importance assessment», *Risk Analysis*, vol. 22, n° 3, p. 579–590.

Saltelli, A. et P. Annoni. 2010, «How to avoid a perfunctory sensitivity analysis», *Environmental Modelling & Software*, vol. 25, n° 12, p. 1508–1517.

Saltelli, A., M. Ratto, T. Andres, F. Campolongo, J. Cariboni, D. Gatelli, M. Saisana et S. Tarantola. 2008, *Global sensitivity analysis : the primer*, John Wiley & Sons.

Saltelli, A., S. Tarantola et K.-S. Chan. 1999, «A quantitative model-independent method for global sensitivity analysis of model output», *Technometrics*, vol. 41, n° 1, p. 39–56.

Santner, T. J., B. J. Williams et W. Notz. 2003, *The design and analysis of computer experiments*, Springer series in statistics, Springer, ISBN 0-387-95420-1.

Schwefel, H.-P. 1981, *Numerical Optimization of Computer Models*, John Wiley & Sons, Inc., ISBN 0471099880.

Shi, Y. et R. Eberhart. 1998a, «A modified particle swarm optimizer», dans *Evolutionary Computation Proceedings of the 1998 IEEE World Congress on Computational Intelligence, May 4–9 1998, Anchorage (Alaska)*, p. 69–73.

Shi, Y. et R. C. Eberhart. 1998b, «Parameter selection in particle swarm optimization», dans *Evolutionary Programming VII : Proceedings of the 7th International Conference, EP98, Mar. 25–27 1998, San Diego (USA)*, Springer, Berlin, Heidelberg, ISBN 978-3-540-68515-9, p. 591–600.

Shi, Y. et R. C. Eberhart. 1999, «Empirical study of particle swarm optimization», dans *Proceedings of the 1999 Congress on Evolutionary Computation, CEC 99, Jul. 6–9 1999, Washington DC (USA)*, vol. 3.

Shi, Y. et R. C. Eberhart. 2001, «Fuzzy adaptive particle swarm optimization», dans *Proceedings of the 2001 Congress on Evolutionary Computation, May 27–30 2001, Seoul (Korea)*, vol. 1, p. 101–106.

Simpson, T. W., J. Poplinski, P. N. Koch et J. K. Allen. 2001, «Metamodels for computer-based engineering design : survey and recommendations», *Engineering with Computers*, vol. 17, n° 2, p. 129–150.

Sobol', I. 2001, «Global sensitivity indices for nonlinear mathematical models and their monte carlo estimates», *Mathematics and Computers in Simulation*, vol. 55, n° 1-3, p. 271–280.

Sobol', I. M. 1990, «On sensitivity estimation for nonlinear mathematical models», *Matematicheskoe Modelirovanie*, vol. 2, n° 1, p. 112–118.

Socha, K. et M. Dorigo. 2008, «Ant colony optimization for continuous domains», *European Journal of Operational Research*, vol. 185, n° 3, p. 1155–1173.

Sörensen, K. 2015, «Metaheuristics - the metaphor exposed», *International Transactions in Operational Research*, vol. 22, n° 1, p. 3–18.

Storlie, C. B. et J. C. Helton. 2008, «Multiple predictor smoothing methods for sensitivity analysis : Description of techniques», *Reliability Engineering & System Safety*, vol. 93, n° 1, p. 28–54.

Storn, R. et K. Price. 1997, «Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces», *Journal of Global Optimization*, vol. 11, n° 4, p. 341–359.

Sudret, B. 2008, «Global sensitivity analysis using polynomial chaos expansions», *Reliability Engineering & System Safety*, vol. 93, n° 7, p. 964–979.

Sulaiman, N., J. Mohamad-Saleh et A. G. Abro. 2013, «A modified Artificial Bee Colony (JA-ABC) optimization algorithm», dans *Proceedings of the International Conference on Applied Mathematics and Computational Methods in Engineering, July 16–19 2013, Rhodes island (Greece)*, p. 74–79.

Surjanovic, S. et D. Bingham. «Virtual Library of Simulation Experiments», http://www.sfu.ca/~ssurjano/index.html. Janvier 2015.

Taillard, E. 2002, «Principes d'implémentation des métaheuristiques», dans *Optimisation approchée en recherche opérationnelle*, édité par J. Teghem et collab., Lavoisier, Paris, p. 57–79.

Takahashi, R., J. Ramirez, J. Vasconcelos et R. Saldanha. 2001, «Sensitivity analysis for optimization problems solved by stochastic methods», *IEEE Tansactions on Magnetics*, vol. 37, n° 5, p. 3566–3569.

Tanabe, R. et A. Fukunaga. 2013, «Success-history based parameter adaptation for differential evolution», dans *Proceedings of the 2013 IEEE Congress on Evolutionary Computation, CEC 2013, Jun. 20–23 2013, Cancun (Mexico)*, p. 71–78.

Tanabe, R. et A. S. Fukunaga. 2014, «Improving the search performance of shade using linear population size reduction», dans *Proceedings of the 2014 IEEE Congress on Evolutionary Computation, CEC 2014, Jul. 6–11 2014, Beijing (China)*, p. 1658–1665.

Tang, L., Y. Dong et J. Liu. 2015, «Differential evolution with an individual-dependent mechanism», *IEEE Transactions on Evolutionary Computation*, vol. 19, n° 4, p. 560–574.

Tissot, J.-Y. et C. Prieur. 2012, «Bias correction for the estimation of sensitivity indices based on random balance designs», *Reliability Engineering & System Safety*, vol. 107, p. 205–213.

Trelea, I. C. 2003, «The particle swarm optimization algorithm : convergence analysis and parameter selection», *Information Processing Letters*, vol. 85, n° 6, p. 317 – 325.

Tseng, L.-Y. et C. Chen. 2008, «Multiple trajectory search for large scale global optimization», dans *Proceedings of the 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence), Jun. 01–06 2008, Hong-Kong (China)*, IEEE, p. 3052–3059.

Villa-Vialaneix, N., M. Follador, M. Ratto et A. Leip. 2012, «A comparison of eight metamodeling techniques for the simulation of N_2O fluxes and N leaching from corn crops», *Environmental Modelling & Software*, vol. 34, p. 51–66.

Wang, H., Z. Wu, X. Zhou et S. Rahnamayan. 2013, «Accelerating artificial bee colony algorithm by using an external archive», dans *Proceedings of the 2003 IEEE Congress on Evolutionary Computation, Dec. 8–12 2003, Canberra (Australia)*, IEEE, p. 517–521.

Wang, Y., Z. Cai et Q. Zhang. 2011, «Differential evolution with composite trial vector generation strategies and control parameters», *IEEE Transactions on Evolutionary Computation*, vol. 15, n° 1, p. 55–66.

Wang, Y.-X. et Q.-L. Xiang. 2008, «Particle swarms with dynamic ring topology», dans *Proceedings of the 2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence), Jun. 01–06 2008, Hong-Kong (China)*, p. 419–423.

Wright, A. H. 1991, «Genetic algorithms for real parameter optimization», dans *Foundations of Genetic Algorithms*, Morgan Kaufmann, p. 205–218.

Wu, G., R. Mallipeddi, P. Suganthan, R. Wang et H. Chen. 2016, «Differential evolution with multi-population based ensemble of mutation strategies», *Information Sciences*, vol. 329, p. 329–345. Special issue on Discovery Science.

Xu, X., Y. Tang, J. Li, C. Hua et X. Guan. 2015, «Dynamic multi-swarm particle swarm optimizer with cooperative learning strategy», *Applied Soft Computing*, vol. 29, p. 169–183.

Yang, X.-S. 2010, «Cuckoo search», dans *Nature-inspired metaheuristic algorithms*, Luniver press, p. 105–116.

Yang, X.-S. et S. Deb. 2009, «Cuckoo search via Lévy flights», dans *Proceedings of the World Congress on Nature & Biologically Inspired Computing, NABIC 2009, Dec. 9–11 2009, Coimbatore (India)*, IEEE, p. 210–214.

Zaharie, D. 2009, «Influence of crossover on the behavior of differential evolution algorithms», *Applied Soft Computing*, vol. 9, n° 3, p. 1126–1138.

Zhang, J. et A. C. Sanderson. 2009, «Jade : Adaptive differential evolution with optional external archive», *IEEE Transactions on Evolutionary Computation*, vol. 13, n° 5, p. 945–958.

Zheng, Y.-L., L.-H. Ma, L.-Y. Zhang et J.-X. Qian. 2003, «On the convergence analysis and parameter selection in particle swarm optimization», dans *Proceedings of the 2003 International Conference on Machine Learning and Cybernetics, Nov. 2–5 2003, Xi'an (China)*, vol. 3, p. 1802–1807 Vol.3.

Zhu, G. et S. Kwong. 2010, «Gbest-guided artificial bee colony algorithm for numerical function optimization.», *Applied Mathematics and Computation*, vol. 217, n° 7, p. 3166–3173.