N° Ordre...../FHC/UMBB/2019

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE M'HAMED BOUGARA - BOUMERDES -

Faculté des Hydrocarbures et de la Chimie

Laboratoire d'Automatique Appliquée

Thèse de Doctorat

Présentée par :

HARFOUCHI Fatima

En vue de l'obtention du **DOCTORAT**

Filière : Génie électrique
Option : Commande automatique

Contribution à l'optimisation par colonies d'abeilles artificielles : Application à la conception de systèmes intelligents

---

Soutenue publiquement le 12 mars 2019 devant le jury :

KESRAOUI Mohamed	Professeur	Université de Boumerdès	Président
RAHMOUNE Fayçal	Professeur	Université de Boumerdès	Examinateur
MAIDI Ahmed	Professeur	Université de Tizi-Ouzou	Examinateur
HAMICHE Hamid	MC/A	Université de Tizi-Ouzou	Examinateur
HABBI HACENE	Professeur	Université de Boumerdès	Encadreur

Année Universitaire : 2018/2019

Remerciements

Je tiens tout d’abord à remercier Allah le tout-puissant, le miséricordieux, pour m’avoir donné la force et la patience d’arriver à finaliser cette thèse.

Je tiens à exprimer toute ma gratitude et ma reconnaissance à mon directeur de thèse Mr. HABBI HACENE, Professeur à l'université de Boumerdès, d'avoir accepté de diriger ce travail et d'avoir mis à ma disposition tous les moyens qui m'ont permis de mener à terme cette étude et aussi pour ses précieux conseils et ses encouragements.

Je remercie chaleureusement Mr. KESRAOUI Mohamed Professeur à l'université de Boumerdès, pour m'avoir fait l'immense honneur de présider le jury de ma soutenance.

Je remercie également, Mr. RAHMOUNE Fayçal, Professeur à l'université de Boumerdès, Mr. MAIDI Ahmed, Professeur à l'université de Tizi-Ouzou, et Mr. HAMICHE Hamid, Maître de conférence à l'université de Tizi-Ouzou pour avoir accepté de rapporter cette thèse. Je leur exprime toute ma reconnaissance pour l'intérêt porté à ce travail, leurs remarques apporteront sûrement matière à ma réflexion.

Dédicaces

Je dédie cette thèse

A ma mère,

Pour son grand amour, son affectation, sa patience et son encouragement pendant les épreuves difficiles ainsi que ses prières qui m'apportent bonheur et réussite.

A mon père,

A mon frère,

Qui sont toujours à mes côtés, prêts à m'aider.

A toute ma famille,

A tous mes amis,

A tous ceux qui me sont chers.

Résumé

Il existe un bon nombre de problèmes scientifiques et techniques qui se trouvent formulés sur la base de concepts d'optimalité, nécessitant par conséquent l'emploie d'une ou plusieurs méthodes d'optimisation pour les résoudre. La thématique abordée dans cette thèse est étroitement liée à la problématique d'optimisation avec l'objectif d'apporter quelques contributions à l'optimisation par méthodes approchées dédiées aux problèmes difficiles de grande dimension. L'intérêt est exclusivement porté sur les métaheuristiques de l'intelligence en essaim qui renferment une série d'approches de recherche globale à base de population de solutions. L'algorithme de colonies d'abeilles artificielles (ABC) est l'une des métaheuristiques qui s'est vue développer une panoplie de variantes appliquées avec succès à différents problèmes scientifiques. Cette récente approche de l'intelligence en essaim s'est montrée jusque-là imposante, en raison d'abord de la simplicité de sa structure algorithmique et puis au faible nombre de paramètres de contrôle qu'elle utilise. Cependant, et comme toute autre métaheuristique, le modèle ABC n'échappe pas à certaines insuffisances d'ordre computationnel. Le compromis entre exploitation et exploration représente en fait une des contraintes auxquelles est confrontée toute étude de développement d'approche d'optimisation à recherche stochastique. La contribution de la thèse s'inscrit justement au centre de cette problématique. Ainsi, nous nous sommes fixés l'objectif d'étoffer le concept ABC de base par le développement d'une nouvelle approche d'optimisation par essaims d'abeilles à apprentissage coopératif, désignée par CLABC, qui se fond sur une caractérisation comportementale assez poussée des essaims d'abeilles lors du fourragement. La modélisation des concepts comportementaux introduits est effectuée au sens de l'amélioration des capacités de diversification et d'intensification du processus évolutif de recherche globale. L'approche d'optimisation proposée est validée d'abord sur des problèmes d'optimisation numériques de différentes dimensionnalités, puis sur des problèmes typiques relevant du domaine de la conception des systèmes intelligents et de l'ingénierie des systèmes. Les contributions apportées le long de cette thèse avancent un outil d'optimisation très prometteur qui se rajoute par excellence à la variété de méthodes métaheuristiques déjà existantes.

Mots clés : Optimisation, Problèmes d'optimisation difficiles, Métaheuristiques, Algorithme de colonies d'abeilles artificielles, Intelligence en essaim.

Abstract

Many scientific and technical problems are formulated on the basis of optimality concepts, which implies the use of one or more optimization methods to solve them. The subject under study in this doctoral thesis is closely related to the field of mathematical optimization with the aim to introduce specific contributions to the class of approximate optimization methods which are devoted to high-dimensional problems. On this issue, a special emphasis is dedicated to metaheuristics which belong to the class of swarm intelligence techniques. Artificial bee colony optimization strategy (ABC) is one of the most impressive metaheuristic methods that has been successfully applied to a variety of complex scientific problems. This recent approach of swarm intelligence has so far demonstrated high performance computing, with regard to the simplicity of its algorithmic framework and to the few control parameters it uses. However, the ABC model as many other population-based approaches has some computational weakness. The trade-off between exploitation and exploration is in fact one of the main constraints that have important impact on the convergence of the search process. The contribution of the present thesis is precisely concerned with this challenging issue. To deal with, we propose a novel bee swarm optimization approach, referred to as cooperative learning artificial bee colony (CLABC), whose design concept is built upon a novel behavioral characterization of collective foraging strategy of bee swarms. Modeling of the introduced behavioral concepts is conducted at different levels in order to improve the diversification and intensification abilities of the evolving search process of the proposed intelligent optimization method. To assess the computational performance of the proposed CLABC model, complex numerical optimization problems of different dimensionalities have been considered in the simulation study which is also performed on typical scientific and engineering problems with application to intelligent systems design. With regard to the obtained results, it can be concluded that the contributions made along this doctoral thesis have successfully introduced a very promising intelligent optimization tool which has been shown to perform considerably well among many other advanced population-based methods.

Keywords : Optimization, Hard optimization problem, Metaheuristics, Artificial bee colony algorithm, Swarm intelligence.

ملخص

هناك العديد من المشاكل العلمية والتقنية التي تتم صياغتها كمشاكل تحسين، والتي تتطلب استخدام طريقة تحسين أو أكثر لحلها. يرتبط محور هذه الرسالة ارتباطًا وثيقًا بمشكلة التحسين وذلك بغية تقديم بعض المساهمات للتحسين من الطرق التقريبية المخصصة لحل المشكلات الصعبة ذات الأبعاد الكبيرة. حيث نركز اهتمامنا حصريًا على الخوارزميات القائمة على ذكاء الأسراب التي تحتوي على سلسلة من مناهج البحث العالمية القائمة على مجموعة من الحلول. خوارزمية مستعمرات النحل الاصطناعية (ABC) هي واحدة من الخوارزميات التي تم تطويرها حديثا ، حيث انه توجد مجموعة هائلة من النسخ التي تم تطبيقها بنجاح على مختلف المشاكل العلمية. وقد أظهرت هذه الطريقة القائمة على ذكاء الأسراب إلى الآن نتائج عدة مثيرة للإهتمام ، ويرجع ذلك أولا إلى بساطة تركيبتها الخوارزمية ثم إلى إعدادات تحكمها القليلة. ومع ذلك ، فمثل جميع الخوارزميات القائمة غلى ذكاء الأسراب ، لا يفلت التحسين باستعمال مستعمرات النحل الاصطناعية من بعض أوجه القصور الحسابية. إذ تعتبر الموازنة بين الاستغلال والاستكشاف أمرا ضروريا لتحسين البحث العشوائي الأمثل. تتمحور هذه الأطروحة حول المساهمة في حل الإشكالية المذكورة. حيث تهدف هذه الدراسة إلى توسيع مفهوم ABC الأساسي من خلال تطوير نهج جديد قائم على التعلم التعاوني لأسراب النحل CLABC ، والذي يعتمد على الوصف السلوكي الشامل لأسراب النحل أثناء قيامهم بالبحث عن الطعام. لقد تمت صياغة المفاهيم السلوكية المقدمة لهدف تحسين التنويع والتكثيف لآلية البحث التطورية العالمية. يتم أولاً التحقق من فعالية نهج التحسين المقترح عبر مشاكل التحسين العددية ذات الأبعاد المختلفة ، ثم عبر بعض مشكلات مجال تصميم الأنظمة الذكية وهندسة الأنظمة النموذجية. الهدف من هذا العمل هو إقتراح أداة تحسين فعالة تضاف بامتياز إلى مجموعة خوارزميات البحث العالمية.

الكلمات المفتاحية : تحسين، مشاكل التحسين الصعبة، الطرق التقريبية العامة، خوارزمية مستعمرات النحل الاصطناعية، ذكاء الأسراب.

Table des matières

Liste des figures v
Liste des tableaux vii
Liste des algorithmes x
Notations et Abréviations xi
Introduction générale 1
Chapitre I : État de l'art des outils de l'optimisation mathématique 5
I.1 Introduction 5
I.2 L'optimisation mathématique 5
I.2.1 Problème d'optimisation 6
I.2.2 Notions et concepts relatifs à l'optimisation 6
I.3 Optimisation difficile 7
I.3.1 Optimisation difficile à variables discrètes 8
I.3.2 Optimisation difficile à variables continues 9
I.4 Les méthodes d'optimisation 9
I.4.1 Les méthodes exactes 9
I.4.2 Les méthodes approchées 10
I.5 Les métaheuristiques à base de population de solutions 11
I.5.1 Les algorithmes évolutionnaires 12
I.5.1.1 Algorithme à évolution différentielle 12
I.5.1.2 Algorithme génétique 14
I.5.2 L'intelligence en essaim 16
I.5.2.1 Optimisation par colonies de fourmis 16
I.5.2.2 Optimisation par essaims de particules 19
I.5.2.3 Optimisation par colonies d'abeilles 20
I.6 Applications de l'optimisation en ingénierie des systèmes 21
I.7 Conclusion 22
Chapitre II : Optimisation par colonies d'abeilles artificielles 23
II.1 Introduction 23
II.2 L'essaim d'abeilles 24

II.3 Modèle comportemental de recherche de nourriture des essaims d'abeilles 25
II.3.1 Caractérisation du modèle 25
II.3.2 Mécanisme de communication 26
II.4 L'algorithme de colonies d'abeilles artificielles 26
II.4.1 Modèle de base de l'algorithme ABC 28
II.4.2 Les variantes de l'algorithme ABC 30
II.5 Applications de l'algorithme ABC 33
II.6 Problématique et contraintes 34
II.6.1 Cadre d'évaluation 34
II.6.2 Critère d'évaluation 36
II.6.3 Critère d'arrêt 36
II.6.4 Mise en œuvre et limitations 37
II.7 Conclusion 37
Chapitre III : Nouvelle approche d'optimisation par essaims d'abeilles à apprentissage coopératif 39
III.1 Introduction 39
III.2 La stratégie d'optimisation proposée 40
III.2.1 L'algorithme CLABC 41
III.2.1.1 Structure et mécanisme de recherche 41
III.2.1.2 Procédure de recherche 42
III.2.2 L'algorithme mCLABC 42
III.2.2.1 Structure et mécanisme de recherche 44
III.2.2.2 Procédure de recherche 45
III.2.2.3 Equilibre entre exploitation et exploration 47
III.3 Analyse de performance de la stratégie d'optimisation proposée 49
III.3.1 Problèmes d'optimisation de fonctions numériques standards 50
III.3.1.1 Comparaison au modèle ABC de base 51
III.3.1.2 Comparaison aux variantes du modèle ABC de base 56
III.3.1.3 Analyse statistique de performance de la stratégie proposée 57
III.3.2 Problèmes d'optimisation de fonctions complexes 61
III.3.2.1 Application aux fonctions composites CEC'05 62
III.3.2.1.1 Résultats comparatifs des variantes du modèle ABC 62
III.3.2.1.2 Résultats comparatifs des variantes du modèle DE 63
III.3.2.2 Application aux fonctions composites CEC'13 64

III.4 Analyse de robustesse de la stratégie d'optimisation proposée 67
III.4.1 Impact du partitionnement des essaims d'abeilles 67
III.4.2 Impact du processus d'altération des sources de nourriture 68
III.4.3 Impact du mécanisme de recherche 69
III.5 Conclusion 70
Chapitre IV : Étude de cas d'applications 72
IV.1 Introduction 72
IV.2 Identification des systèmes neuro-flous 72
IV.2.1 Architecture du réseau neuro-flou adaptatif (ANFIS) 73
IV.2.2 Apprentissage du réseau ANFIS par le modèle mCLABC 75
IV.2.4 Résultats d'application 76
IV.3 Conception des classifieurs neuronaux 76
IV.3.1 Les réseaux de neurones artificiels 78
IV.3.2 Apprentissage des classifieurs neuronaux par l'algorithme mCLABC 80
IV.3.2.1 Problèmes benchmarks de moyenne dimension 80
IV.3.2.2 Problèmes benchmarks de grande dimension 83
IV.4 Classification de données non supervisée 86
IV.4.1 Formulation du problème 86
IV.4.2 Clustering à base du modèle mCLABC 87
IV.5 Conception mécanique sous contraintes 90
IV.5.1 Adaptation de la structure algorithmique de l'algorithme mCLABC aux problèmes contraints 90
IV.5.2 Application aux problèmes de conception mécanique 91
IV.5.3 Résultats d'application 94
IV.6 Conception d'un compensateur de non linéarité pour le transducteur LVDT 96
IV.6.1 Le capteur inductif linéaire LVDT 96
IV.6.2 Compensation de non linéarité par réseau FLANN-mCLABC 97
IV.6.3. Résultats d'application 99
IV.7 Optimisation du contrôle PID 103
IV.7.1 Le contrôleur PID 103
IV.7.2 Procédure d'optimisation et résultats 104
IV.8 Conclusion 109
Conclusion générale 110
Bibliographie 113

Annexe A 122
Annexe B 125

Liste des figures

Figure I.1 : Classification des méthodes d'optimisation 10
Figure I.2 : Illustration de l'opérateur de croisement. 15
Figure I.3 : Illustration de l'opérateur de mutation. 15
Figure I.4 : Illustration de la capacité des fourmis à chercher de la nourriture en minimisant leur parcours 17
Figure II.1 : Schéma illustratif du comportement d'une abeille lors du fourragement 27
Figure II.2 : Organigramme de l'optimisation par colonies d'abeilles artificielles. 31
Figure III.1 : Les différentes possibilités de mouvement d'une abeille selon les équations de recherche définies dans le modèle mCLABC dans un espace paramétrique de 2 dimension..... 46
Figure III.2 : Stratégie sélective d'altération des sources de nourriture du modèle mCLABC..... 48
Figure III.3 : Taux de sélection des équations de recherche dans le modèle mCLABC..... 49
Figure III.4 : Les courbes de convergence des algorithmes ABC, CLABC et mCLABC illustrant 15 fonctions numériques de différentes dimensionnalités..... 55
Figure IV.1 : Architecture du modèle ANFIS à deux entrées, une sortie et deux fonctions d'appartenance par entrée..... 74
Figure IV.2 : Performance du modèle mCLABC sur les problèmes d'identification à base du réseau neuro-flou ANFIS 77
Figure IV.3 : Modèle d'un neurone artificiel. 79
Figure IV.4 : Réseau de neurones multicouches à topologie 3-3-2. 79
Figure IV.5 : Problèmes de conception mécanique. 94
Figure IV.6 : Schéma simplifié d'un transducteur LVDT. 97
Figure IV.7 : Structure du modèle FLANN optimisé par un algorithme évolutionnaire. 98
Figure IV.8 : Le LVDT du laboratoire (KL-68004) et son circuit de conditionnement. 99
Figure IV.9 : Les courbes de convergence des compensateurs FLANN-ABC, FLANN-gABC et FLANN-mCLABC développés pour les transducteurs LVDT-1 et LVDT-2..... 101

Figure IV.10 : Compensation de la non-linéarité du LVDT-1 par le modèle ELM (sinus), l'algorithme DE et le modèle GA-ANN.....101
Figure IV.11 : Compensation de la non-linéarité du LVDT-1 et du LVDT-2 par le modèle FLANN-ABC.....102
Figure IV.12 : Compensation de la non-linéarité du LVDT-1 et du LVDT-2 par le modèle FLANN-gABC.....102
Figure IV.13 : Compensation de la non-linéarité du LVDT-1 et du LVDT-2 par le modèle FLANN-mCLABC.....102
Figure IV.14 : Principe de paramétrage d'un contrôleur PID à base de métaheuristique.....104
Figure IV.15 : La réponse indicielle du processus 1 associée à la meilleure solution obtenue par les méthodes ZN, HS, IBA, ABC et mCLABC.....107
Figure IV.16 : La réponse indicielle du processus 1 associée à la mauvaise solution obtenue par les méthodes HS, IBA, ABC et mCLABC.....107
Figure IV.17 : La réponse indicielle du processus 2 associée à la meilleure solution obtenue par les méthodes ZN, HS, IBA, ABC et mCLABC.....107
Figure IV.18 : La réponse indicielle du processus 2 associée à la mauvaise solution obtenue par les méthodes HS, IBA, ABC et mCLABC.....108
Figure IV.19 : La réponse indicielle du processus 3 associée à la meilleure solution obtenue par les méthodes ZN, HS, IBA, ABC et mCLABC.....108
Figure IV.20 : La réponse indicielle du processus 3 associée à la mauvaise solution obtenue par les méthodes HS, IBA, ABC et mCLABC.....108

Liste des tableaux

Tableau III.1 : Comparaison des performances des algorithmes mCLABC, CLABC et ABC sur des problèmes d'optimisation des fonctions F1-F20 (D = 15) et des fonctions F21-F22 (D = 50). 53
Tableau III.2 : Comparaison des performances des algorithmes mCLABC, CLABC et ABC sur des problèmes d'optimisation des fonctions F1-F20 (D = 30) et des fonctions F21-F22 (D = 100). 53
Tableau III.3 : Comparaison des performances des algorithmes mCLABC, CLABC et ABC sur des problèmes d'optimisation des fonctions F1-F20 (D = 60) et des fonctions F21-F22 (D = 200). 54
Tableau III.4 : Comparaison des performances des algorithmes mCLABC, ABCVSS, MuABC, OGABC, ABCbest et CABC sur des problèmes d'optimisation des fonctions F1-F20 (D = 15) et des fonctions F21-F22 (D = 50). 58
Tableau III.5 : Comparaison des performances des algorithmes mCLABC, ABCVSS, MuABC, OGABC, ABCbest et CABC sur des problèmes d'optimisation des fonctions F1-F20 (D = 30) et des fonctions F21-F22 (D = 100). 59
Tableau III.6 : Comparaison des performances des algorithmes mCLABC, ABCVSS, MuABC, OGABC, ABCbest et CABC sur des problèmes d'optimisation des fonctions F1-F20 (D = 60) et des fonctions F21-F22 (D = 200). 60
Tableau III.7 : Analyse statistique de la performance de la stratégie mCLABC proposée sur les problèmes d'optimisation des fonctions numériques F1-F20 (D = 15) et F21-F22 (D = 50). 61
Tableau III.8 : Analyse statistique de la performance de la stratégie mCLABC proposée sur les problèmes d'optimisation des fonctions numériques F1-F20 (D = 30) et F21-F22 (D = 100). 61
Tableau III.9 : Analyse statistique de la performance de la stratégie mCLABC proposée sur les problèmes d'optimisation des fonctions numériques F1-F20 (D = 60) et F21-F22 (D = 200). 62
Tableau III.10 : Comparaison des performances du modèle mCLABC sur les fonctions composites CEC'05 de dimension 10. 63

Tableau III.11 : Comparaison des performances du modèle mCLABC sur les fonctions composites CEC'05 de dimension 30.....64
Tableau III.12 : Comparaison des performances du modèle mCLABC sur les fonctions composites CEC'13 de dimension 50.....65
Tableau III.13 : Comparaison des performances du modèle mCLABC sur les fonctions composites CEC'13 de dimension 100.....66
Tableau III.14 : Analyse de robustesse du modèle mCLABC vis-à-vis du partitionnement des essaims d'abeilles.....68
Tableau III.15 : Analyse de robustesse du modèle mCLABC vis-à-vis du procédé d'altération des caractéristiques de la nourriture.....69
Tableau III.16 : Analyse de robustesse du modèle mCLABC vis-à-vis du mécanisme de recherche à règles multiples.....70
Tableau IV.1 : Résultats d'application de l'approche mCLABC à l'apprentissage du réseau ANFIS 76
Tableau IV.2 : Paramètres des problèmes de classification benchmarks de moyenne dimension.....81
Tableau IV.3 : Résultats d'application de l'approche mCLABC à la classification neuronale des problèmes XOR, 3Bit parity et encoder/decoder.....82
Tableau IV.4 : Paramètres des problèmes benchmarks de grande dimension (real-world data).....84
Tableau IV.5 : Résultats d'application de l'approche mCLABC à la classification neuronale des données Cancer, Diabetes et Heart.....85
Tableau IV.6 : Paramètres des problèmes de classification.....88
Tableau IV.7 : Résultats comparatifs de classification de données non supervisée (clustering).....89
Tableau IV.8 : Résultats comparatifs des erreurs de classification (EP).....89
Tableau IV.9 : Les paramètres de contrôle des algorithmes ABC, UABC et mCLABC.....95
Tableau IV.10 : Résultats statistiques d'application de l'approche mCLABC proposée aux problèmes de conception mécanique avec contraintes.....95
Tableau IV.11 : Comparaison des performances du modèle FLANN-mCLABC avec d'autres approches avancées sur le problème de compensation du LVDT-1.....100
Tableau IV.12 : Comparaison des performances du modèle FLANN-mCLABC avec d'autres approches avancées sur le problème de compensation du LVDT-2.....100
Tableau IV.13 : Poids et plages de variation des paramètres de réglage.....105

Tableau IV.14 : Résultats comparatifs des erreurs moyennes et des écart-types obtenus par les algorithmes ZN, HS, BA, IBA, ABC et mCLABC.....105
Tableau IV.15 : Performances du réglage PID obtenues pour la meilleure solution du problème d'optimisation.....106
Tableau IV.16 : Performances du réglage PID obtenues pour la moyenne solution du problème d'optimisation.....106
Tableau IV.17 : Performances du réglage PID obtenues pour la mauvaise solution du problème d'optimisation.....106

Liste des algorithmes

Algorithme I.1 : Pseudo-code d'optimisation à évolution différentielle 13
Algorithme I.2 : Pseudo-code d'un algorithme génétique 15
Algorithme I.3 : Pseudo-code d'optimisation par colonies de fourmis 18
Algorithme I.4 : Pseudo-code d'optimisation par essaim de particules 20
Algorithme II.1 : Structure générale de l'algorithme ABC.....28
Algorithme II.2 : Pseudo-code d'optimisation par colonies d'abeilles artificielles 30
Algorithme III.1 : Pseudo-code de l'algorithme CLABC 43
Algorithme III.2 : Pseudo-code de l'algorithme mCLABC..... 50
Algorithme III.3 : Pseudo-code du processus de recherche de nourriture (fourragement) associé aux sous-groupes d'abeilles Employed dans le modèle mCLABC..... 51
Algorithme III.4 : Pseudo-code du processus de recherche de nourriture (fourragement) associé aux sous-groupes d'abeilles Onlooker dans le modèle mCLABC 52

Notations et Abréviations

ABC : Artificial Bee Colony (Optimisation par colonies d'abeilles artificielles).

ACO : Ant Colony Optimization (Optimisation par colonies de fourmis).

ANFIS : Adaptive Neuro-Fuzzy Inference System (Système d'inférence neuro-flou adaptatif).

ANN : Artificial Neural Network (Réseau de neurones artificiels).

BA : Bat Algorithm (Algorithme de chauve-souris).

CEC : Congress on Evolutionary Computation (Congrès sur le calcul évolutionnaire).

CLABC : Cooperative Learning Artificial Bee Colony (Colonies d'abeilles artificielles à apprentissage coopératif).

CS : Cuckoo Search (La recherche coucou).

DE : Differential Evolution (Evolution différentielle).

EA : Evolutionary Algorithm (Algorithme Evolutionnaire).

FLANN : Functional Link Artificial Neural Network (Réseau de neurones artificiels à liens fonctionnels).

GA : Genetic Algorithm (Algorithme génétique).

LVDT : Linear Variable Displacement Transducer (Transducteur de déplacement inductif à transformateur différentiel).

mCLABC : modified Cooperative Learning Artificial Bee Colony (Colonies d'abeilles artificielles à apprentissage coopératif modifié).

PID : Proportionnelle, Intégrale et Dérivée.

PSO : Particle Swarm Optimization (Optimisation à essaims de particules).

SI : Swarm Intelligence (Intelligence en essaim).

$\boldsymbol{F}$ **:** Coefficient de mutation.

$\boldsymbol{CR}$ **:** Crossover Rate (Coefficient de croisement).

$\boldsymbol{\tau_{ij}(t)}$ **:** Densité de la trace de phéromone.

$\boldsymbol{\rho}$ **:** Paramètre d'évaporation de phéromone.

$\boldsymbol{MCN}$ **:** Maximum Cycle Number (Nombre maximal de cycles).

$\boldsymbol{\eta_{ij}}$ **:** Coefficient de visibilité de la fourmi.

$\boldsymbol{w_{ij}}$ **:** Poids synaptiques qui relie le neurone i avec le neurone j.

$\boldsymbol{D}$ **:** Dimension du problème d'optimisation.

$\boldsymbol{fit}$ **:** Fonction d'adéquation (fitness).

$\boldsymbol{x_{min}^{j}}$ **:** Limite inférieure de l'espace de recherche de la dimension j.

$\boldsymbol{x_{max}^{j}}$ **:** Limite supérieure de l'espace de recherche de la dimension j.

$\boldsymbol{\varphi_{i}^{j}}$ **:** Nombre aléatoire généré pour chaque dimension dans l'intervalle $[0,1]$.

$\emptyset_{i}^{j}$ **:** Nombre aléatoire généré pour chaque dimension dans l'intervalle $[-1,1]$.

$\boldsymbol{x_{mean}^{j}}$ **:** Solution moyenne correspondant à la $j^{ième}$ dimension.

$\boldsymbol{x_{best}^{j}}$ **:** La $j^{ième}$ dimension de la meilleure solution globale obtenue.

$\boldsymbol{p_i}$ **:** Probabilité de sélection d'une solution i par une abeille Onlooker.

$\boldsymbol{x_i}$ **:** Solution courante à l'itération *i*.

$\boldsymbol{v_i}$ **:** Nouvelle solution potentielle à l'itération *i*..

$\boldsymbol{iter}$ **:** L'itération courante.

$\boldsymbol{Max_FEs}$ **:** Maximum number of Function Evaluations (Nombre maximal d'évaluations de la fonction objectif).

$\boldsymbol{Des(z_i, c_j)}$ **:** Distance entre le vecteur d'objets z_i et les centres des clusters c_j.

$\boldsymbol{g_j(x)}$ **:** Contrainte d'inégalité.

$\boldsymbol{h_j(x)}$ **:** Contrainte d'égalité.

Introduction générale

Depuis déjà quelques décennies, l'intelligence artificielle, en tant que discipline à part entière, s'est vu développer une multitude de concepts, d'outils et de méthodes fondamentales, qui ont été projetés sur différentes applications relevant du domaine de l'ingénierie des systèmes. Les réalisations basées sur les techniques intelligentes ne manquent cependant pas, et demeurent suffisamment perceptibles tant sur le plan méthodologique que sur le plan expérimental. Ceci a été éventuellement rendu possible grâce à la panoplie d'approches calquées à partir de concepts biologiques naturels, à l'instar des stratégies évolutionnaires, des réseaux neuronaux, des systèmes experts à base de règles et de l'intelligence en essaim. Une telle catégorisation est néanmoins loin d'être exhaustive au regard des grandes avancées que rapportent les travaux de recherche axés sur ce domaine de recherche particulier.

L'effet « Intelligence artificielle » n'a presque épargné aucune discipline scientifique, du moins sur le plan développement et recherche. C'est ainsi que se trouve impactée l'une des plus anciennes disciplines scientifiques qu'est l'optimisation mathématique. Il en ressort alors une branche assez distinguée dite « optimisation intelligente », qui se fond essentiellement sur l'intelligence en essaim, pour cadrer toute approche de recherche globale à base de populations de solutions [51]. Ainsi, les techniques de l'optimisation intelligente sont perçues comme étant des méthodes approchées dédiées principalement aux problèmes difficiles de grande dimension [3]. Néanmoins, elles peuvent toujours s'appliquer à n'importe quelle catégorie de problèmes, faciles ou difficiles, bien ou mal formulés, avec ou sans contraintes.

Sur le plan conceptuel, l'optimisation intelligente renferme une série de métaheuristiques à caractère stochastique, qui font évoluer, au cours du processus itératif de recherche, une ou plusieurs populations de solutions en évaluant leur adéquation dans le but d'obtenir des solutions optimales ou quasi-optimales de qualité. Contrairement aux heuristiques dédiées, qui sont des méthodes d'optimisation approchées spécifiques à un problème donné, les métaheuristiques regroupent des méthodes génériques non dédiées à un problème en particulier et permettent une meilleure exploration des grands espaces de recherche (intensification) de part leur pouvoir de diversification [2]. C'est selon cette perception

générale que se sont développées la plupart des méthodes d'optimisation à essaims d'abeilles, dont l'optimisation par colonies d'abeilles artificielles [22], désignée communément par ABC (Artificial Bee Colony), qui jusque-là s'énonce comme étant l'une des techniques les plus imposantes, de part sa simplicité algorithmique et ses performances supérieures. L'approche ABC constitue justement l'objet d'analyse et de développement de ce travail de thèse.

Le modèle ABC, et comme toute autre méthode d'optimisation intelligente, n'échappe cependant pas à certaines limitations d'ordre computationnel à impact effectif sur la diversification et l'intensification du processus de recherche, qui sont à la base de toute évolution optimale de solutions dans les modèles algorithmiques de recherche globale [55]. En effet, une meilleure exploration aura pour effet de préserver la diversité de la population et éviter la convergence prématurée, alors qu'une meilleure exploitation aurait à améliorer l'évolution et la qualité des solutions. La problématique majeure répond donc à un besoin d'équilibrage entre exploration et exploitation des solutions potentielles de sorte à gérer efficacement ce compromis, dont l'impact est immédiat sur la convergence et la robustesse du processus de recherche et l'adéquation des solutions obtenues. A cette contrainte majeure se rajoutent le caractère de recherche unidimensionnel du modèle ABC et les règles de voisinage qu'il emploie, qui tout de même apportent une grande souplesse à la structure algorithmique de la méthode, mais en revanche limitent ses capacités d'exploitation.

La contribution de la thèse s'inscrit justement au centre de cette problématique. Ainsi, nous nous sommes fixés l'objectif d'étoffer le concept ABC de base par le développement d'une nouvelle approche d'optimisation par essaims d'abeilles à apprentissage coopératif, que nous convenons de désigner par CLABC (Cooperative Learning Artificial Bee Colony). Pour ce faire, il est d'abord question d'opérer des changements sur le modèle ABC qui ne se limitent guère à un simple réajustement des paramètres de contrôle, mais touchent essentiellement à la structure algorithmique du modèle d'optimisation de base avec l'introduction de nouveaux concepts comportementaux caractéristiques des essaims d'abeilles. A cet égard, la procédure de conception visée passera par la projection sur plateforme algorithmique de la caractérisation comportementale proposée, et qui s'effectue par modélisation des trois processus suivants :

- Le processus de partitionnement des essaims d'abeilles et du fourragement collectif.
- Le processus de recherche de nourriture à règles multiples.
- Le processus d'altération des sources de nourriture.

L'autre objectif escompté concerne l'étude de validation du modèle d'optimisation proposé, qui en fait constitue une étape fondamentale dans le processus de développement de

la nouvelle approche. Pour ce faire, il sera question d'analyser la performance et la robustesse de la nouvelle stratégie sur des problèmes d'optimisation difficile de différentes dimensionnalités, définis sur des fonctions numériques standards et composites catégorisées dans les bases de référence CEC'2005 et CEC'2013. Cette étape de validation sera également complétée par une étude de cas d'applications relevant du domaine des systèmes intelligents et de l'ingénierie des systèmes. Les applications à appréhender se résument à des problèmes typiques formulés en tant que problèmes d'optimisation difficile, et qui concernent la conception de classifieurs neuronaux, l'identification des systèmes neuro-flous, la classification de données non supervisée, la conception mécanique sous contraintes, la compensation de la non-linéarité d'un transducteur inductif à transformateur différentiel (LVDT) et l'optimisation du contrôle PID.

Cette thèse comprend quatre chapitres organisés de la manière suivante:

Le chapitre I passe en revue les différents outils, concepts et méthodes de l'optimisation mathématique. On y aborde en particulier les métaheuristiques à base de population de solutions, notamment les approches évolutionnaires et les approches de l'intelligence en essaim, à travers une description détaillée des principes y afférents, de leurs structures algorithmiques et certains de leurs domaines d'application en ingénierie des systèmes.

Le chapitre II est dédiée à l'optimisation par colonies d'abeilles artificielle. On y retrace le fondement biologique de la métaheuristique, en présentant le modèle comportemental des essaims d'abeilles associé au processus de fourragement chez les abeilles. Le modèle algorithmique de base de l'approche ABC et certaines de ses variantes y sont également explicités. Entre autre, le chapitre revient sur les limitations du processus de recherche du modèle ABC et le cadre d'évaluation de ses performances.

Le chapitre III est consacrée à la principale contribution de la thèse qu'est la nouvelle stratégie d'optimisation par essaims d'abeilles à apprentissage coopératif. Plus précisément, on y retrace les principales démarches de développement, les structures algorithmiques proposées, les mécanismes de recherche, suivis d'une analyse intensive de performance menée à travers des études comparatives impliquant des variantes récentes du modèle ABC et des méthodes métaheuristiques avancées. Le chapitre est complété par une analyse de robustesse de l'approche proposée illustrant l'impact de différents paramètres de conception sur la performance du modèle d'optimisation développé.

Le chapitre IV présente enfin une étude de validation de l'approche d'optimisation par essaims d'abeilles à apprentissage coopératif sur des problèmes scientifiques relevant du domaine des systèmes à intelligence artificielle et du domaine de l'ingénierie des systèmes. A

cet effet, six différentes applications y sont appréhendées, elles concernent l'identification des systèmes neuro-flous, la conception des classifieurs neuronaux, la classification de données non supervisée, la conception mécanique sous contraintes, la compensation de la non-linéarité d'un transducteur inductif (LVDT) et l'optimisation du contrôle PID. Des conclusions sur la faisabilité de l'approche proposée en tant que nouvel outil d'optimisation intelligente y sont ensuite formulées sur la base des résultats comparatifs illustrés.

La thèse se termine par une conclusion générale portant sur les contributions apportées le long de ce travail de recherche et les perspectives à prétendre, suivie de deux annexes.

Chapitre I

État de l'art des outils de l'optimisation mathématique

I.1 Introduction

La recherche d'un état optimal est l'acte fondamental d'un comportement purement naturel qui, par instinct même, se trouve associé à tout agissement en quête de visées propres. C'est ainsi que de nombreux problèmes existentiels peuvent être formulés sous la forme de problèmes d'optimisation simples ou complexes. L'optimisation, en tant que discipline scientifique à part entière, intervient donc sur un problème existentiel de nature quelconque pour l'appréhender en vue de le transformer en un modèle manipulable que nous pouvons étudier afin d'en extraire les propriétés structurelles et de caractériser les solutions recherchées. Ceci traduit en fait la démarche fondamentale d'exploiter cette caractérisation afin de développer les algorithmes de calcul adéquats [1].

D'une manière générale, la résolution d'un problème d'optimisation nécessite d'abord une modélisation du problème réel et une technique de résolution adaptée. Dans ce chapitre, nous présentons les concepts fondamentaux de l'optimisation, nous décrivons ensuite l'optimisation difficile et les problèmes à variables discrètes et à variables continues qui lui sont associés. Un aperçu sur les différentes méthodes d'optimisation y compris les méthodes d'optimisation à base de population de solutions sera ensuite présenté.

I.2 L'optimisation mathématique

Comme nous venons de le souligner, l'objet principal de l'optimisation est la construction d'algorithmes permettant de rechercher dans un espace de solutions potentielles, une solution optimale "meilleure" quantifiée par une fonction objectif qui conduit à vouloir maximiser ou minimiser le problème. Un grand nombre de problèmes peuvent être, en effet, traités de façon qu'on puisse ajuster et améliorer leurs paramètres pour produire des résultats fiables.

I.2.1 Problème d'optimisation

Un problème d'optimisation, peut être défini par le couple (X, f), où X représente un ensemble de solutions réalisables ou possibles (appelé aussi espace de recherche ou espace de décision), et f la fonction objectif à optimiser qui associe à chaque élément x de l'ensemble X un scalaire dans $\mathbb{R}$. Résoudre le problème revient alors à trouver une solution $x^* \in X$ telle que $\forall\, x \in X;\ f(x^*) \leq f(x)$ (problème de minimisation) ou $f(x^* \geq f(x)$ (problème de maximisation). Une telle solution x^* est dite solution optimale, ou optimum [2].

I.2.2 Notions et concepts relatifs à l'optimisation

Nous présentons à présent quelques définitions utilisées couramment dans le domaine de l'optimisation mathématique.

Solution: est une affectation de toutes les variables du problème.

Espace de recherche: il est défini par le domaine dans lequel la recherche d'un optimum va s'effectuer.

Optimum global (solution optimale): une solution $x^* \in X$ est un optimum global si et seulement si :

$\forall\, x \in X\ et\ x^* \neq x$:

$f(x^*) \leq f(x)$ dans le cas d'un problème de minimisation

$f(x^*) \geq f(x)$ dans le cas d'un problème de maximisation

Optimum local: une solution $x^* \in X$ est un optimum local si et seulement si :

$\forall\, x \in v(x^*)$,

$f(x^*) \leq f(x)$ dans le cas d'un problème de minimisation

$f(x^*) \geq f(x)$ dans le cas d'un problème de maximisation

où $v(x^*)$ est le voisinage de x^* et $x^* \neq x$.

Fonction objectif: la fonction objectif (fonction de coût ou fitness) est la traduction en langage mathématique du désir de l'utilisateur. Elle représente le but à atteindre (maximiser ou minimiser la fonction objectif) en utilisant un algorithme d'optimisation. En effet, la qualité d'une solution donnée par un tel algorithme d'optimisation est liée à la valeur de la fonction objectif.

Voisinage: le voisinage $v(x)$ d'une solution x est un sous-ensemble de solutions de X atteignables en modifiant légèrement la solution x.

Contraintes: on appelle contraintes les conditions imposées sur l'espace de recherche que les variables doivent satisfaire. Ces contraintes qui sont souvent sous forme d'égalité ou d'inégalité permettent de limiter l'espace de recherche.

Exploration (diversification): est liée à la capacité de l'algorithme à prospecter les différentes régions inconnues dans l'espace de recherche pour fournir une estimation fiable de l'optimum global.

Exploitation (intensification): se réfère à la capacité d'utiliser les connaissances existantes (les meilleures solutions trouvées) pour déterminer l'optimum global.

I.3 Optimisation difficile

Le degré de difficulté pour résoudre un problème donné avec un algorithme dédié est étroitement lié à sa complexité de calcul, c'est-à-dire, l'existence ou pas d'un algorithme capable de résoudre ce problème en se basant sur une estimation théorique du temps de calcul nécessaire et des besoins en espace mémoire. En effet, le problème d'optimisation est considéré difficile, s'il n'existe aucun algorithme qui n'est capable de le résoudre d'une manière exacte en un temps raisonnable et en utilisant un espace mémoire aussi raisonnable.

Les problèmes d'optimisation difficiles peuvent se diviser en deux catégories: les problèmes discrets et les problèmes continus [3].

Dans le premier cas, les variables sont discrètes et doivent être choisies dans un panel de valeurs précédemment définies. Le fait que le nombre de combinaisons possibles (nombre de solutions réalisables) est en général très élevé rend difficile la détermination d'une meilleure solution en un temps de calcul raisonnable.

Dans le deuxième cas, les variables du problème d'optimisation sont continues et peuvent prendre toutes les valeurs définies sur un intervalle donné (borné ou pas). Les difficultés les plus courantes de ce type de problèmes sont les corrélations non identifiées entre les variables, le caractère bruité des fonctions ou encore des fonctions objectifs qui ne sont accessibles que par dispositif expérimental [4].

La difficulté d'un problème est équivalente à celle du meilleur algorithme proposé pour le résoudre, un problème est facile (traitable) s'il existe un algorithme capable de le résoudre en un temps polynomial, sinon le problème est considéré comme difficile (intraitable) [2].

La théorie de la complexité s'intéresse à l'étude formelle de la difficulté des problèmes en informatique [5]. L'étude de complexité d'un problème consiste en l'étude de la définition même du problème et de sa structure en se basant sur une estimation théorique du temps de calcul et des besoins en espace mémoire afin de conclure si ce problème peut être résolu

efficacement ou pas. Parmi les classes les plus connues, on distingue les deux classes essentielles P et NP [2].

- **La classe de complexité P (Polynomiale) :** elle regroupe l'ensemble des problèmes polynomiaux qui peuvent être résolus par un algorithme déterministe de complexité polynomiale.
- **La classe de complexité NP (Non Polynomiale) :** elle regroupe l'ensemble des problèmes qui peuvent être résolus par un algorithme non déterministe en un temps polynomial. Les problèmes de cette classe peuvent être résolus en énumérant l'ensemble des solutions possibles qui peuvent être évaluées à l'aide d'un algorithme polynomial.

I.3.1 Optimisation difficile à variables discrètes

À première vue, l'optimisation discrète, en tant que sujet appliqué, semble être particulièrement simple. En effet, en présence d'un champ de possibilités, il y a lieu d'examiner chaque option et choisir la meilleure. La difficulté réside dans les chiffres. Les chemins, les tours de voyageur de commerce, les appariements et d'autres objets de base sont tous trop nombreux pour que l'on puisse vérifier les solutions une par une [6].

Cet argument de nombres est facile à voir dans le cas du problème du voyageur de commerce, désigné communément par TSP (Travel Salesman Problem). Supposons que nous devrions visiter "n" villes dans un circuit qui commence et se termine dans la même ville. La longueur d'un tel circuit ne dépend pas du point de départ. Nous pouvons donc choisir n'importe quelle ville comme point de départ. Il en résulte ensuite "$n-1$" choix pour la deuxième ville, "$n-2$" choix pour la troisième ville, etc. Le nombre total de visites équivaut donc à :

$$(n-1)! = (n-1) \times (n-2) \times (n-3) \times \ldots \times 2 \times 1.$$

Pour dix villes, cela ne représente que 362 880 possibilités et il serait assez facile de les examiner toutes à l'aide d'un calculateur. Même vingt villes semblent résolues en passant par tous les $1{,}2 \times 10^{17}$ candidats, si l'on disposerait d'un calculateur assez puissant. Mais aller jusqu'à trente villes donnerait $8{,}8 \times 10^{30}$ visites, ce qui est hors de portée des plateformes informatiques disponibles actuellement. La complexité des calculs dans ce genre de situations est donc effective, mais fondamentalement très contraignante car la recherche de l'optimum dans un temps "raisonnable" est un défi de calcul en lui-même [6].

Des problèmes d'optimisation discrets se posent naturellement dans de nombreux domaines d'applications, notamment la planification, le transport, la fabrication, la finance et la

logistique. Le problème du voyageur de commerce est l'un des problèmes les plus étudiés dans la littérature et qui consiste à trouver le chemin le plus court pour relier un certain nombre de villes. On trouve aussi le problème d'affectation quadratique qui consiste à placer "n" unités communicantes entre elles sur "n" sites prédéterminés, ce problème possède de nombreuses applications telles que la synthèse d'images, la répartition de bâtiments ou de services.

I.3.2 Optimisation difficile à variables continues

Les problèmes d'optimisation difficile à variables continues peuvent être catégorisés en trois classes [7] :

- **Problèmes d'optimisation purs :** plusieurs problèmes à variables continues exigent une étape d'optimisation, un exemple typique est l'optimisation des performances d'un circuit électronique qui consiste à trouver la valeur numérique nécessaire à affecter à chacun de ces paramètres disponibles de sorte que le circuit puisse réaliser un objectif donné autant que possible.
- **Problèmes d'identification ou de caractérisation :** la majorité des problèmes d'identification ou de caractérisation peuvent être traités comme des problèmes d'optimisation difficiles de sorte que les variables à optimiser sont de nature continue. Un exemple typique de cette catégorie est le réglage du modèle d'un moteur électrique synchrone avec une fonction objectif à minimiser basée sur la différence entre les données expérimentales et les calculs basés sur le modèle théorique.
- **Problèmes d'apprentissage :** ce type de problèmes est rencontré généralement dans les réseaux de neurones ou les bases de règles floues. Un problème typique est l'apprentissage du réseau de neurones qui peut s'exprimer comme un problème d'optimisation dont le but est d'optimiser les coefficients synaptiques du réseau.

I.4 Les méthodes d'optimisation

La grande diversité des problèmes d'optimisation a conduit à une grande variété de méthodes de résolution. Ces méthodes peuvent être réparties en deux grandes catégories : les méthodes exactes et les méthodes approchées [2], comme montré en Figure I.1.

I.4.1 Les méthodes exactes

Les méthodes d'optimisation exactes (déterministes) ont l'avantage de fournir des solutions dont l'optimalité est garantie dans un temps de calcul acceptable. Le principe de ces méthodes

consiste généralement à examiner de manière implicite l'ensemble des combinaisons de l'espace de recherche. Elles sont dédiées à certaines formes de problèmes auxquels sont associés des fonctions objectifs convexes, continues ou encore dérivables.

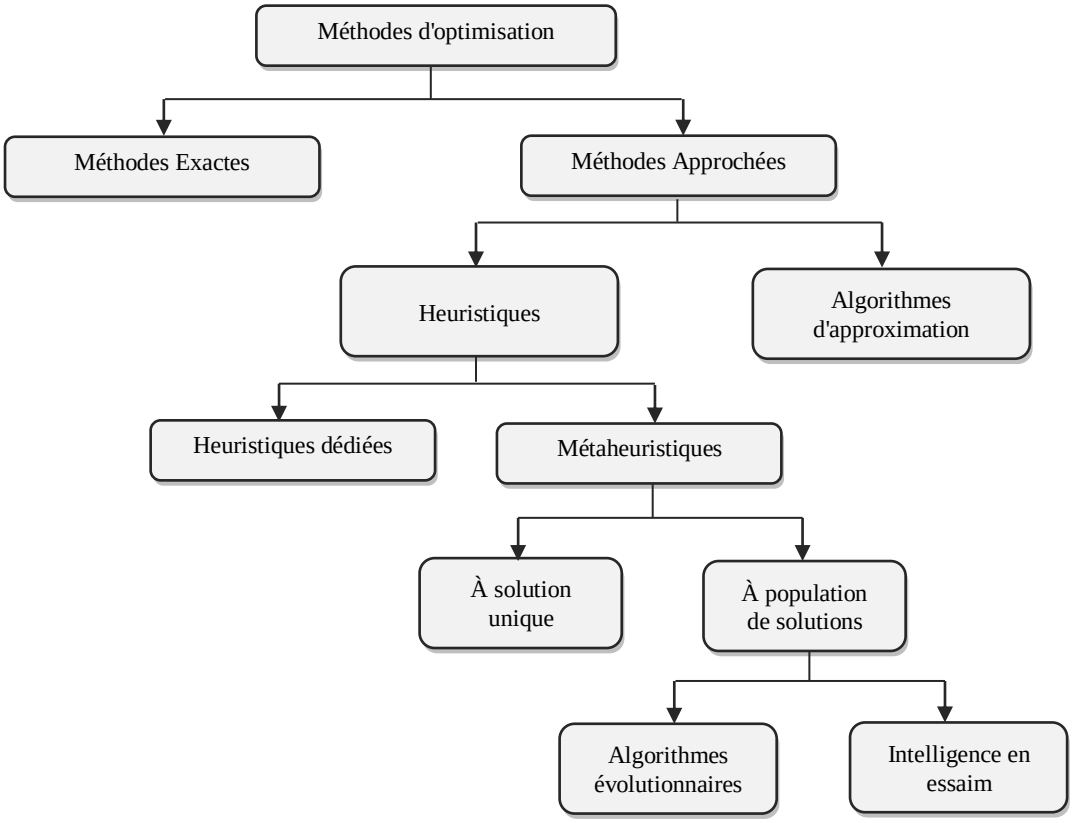


Figure I.1 : Classification des méthodes d'optimisation (inspirée de [2])

Les méthodes exactes sont efficaces pour les problèmes de taille raisonnable, néanmoins le temps de calcul nécessaire pourrait augmenter exponentiellement en fonction de la taille du problème à résoudre. Parmi les nombreuses méthodes exactes, on peut citer la programmation dynamique, l'algorithme A^*, les algorithmes de séparation et évaluation (Branch and Bound), et bien d'autres [2].

I.4.2 Les méthodes approchées

Les méthodes approchées constituent une alternative aux méthodes exactes et s'appliquent principalement aux problèmes de grande dimension ou à des problèmes dont on ne connait

pas d'algorithmes de résolution en temps polynomial et pour lesquels on cherche à déterminer une solution optimale ou quasi-optimale. Les méthodes approchées peuvent s'appliquer à n'importe quelle catégorie de problèmes, faciles ou difficiles, bien ou mal formulés, avec ou sans contraintes. Ces méthodes sont souvent classées en deux catégories : les méthodes d'approximation et les heuristiques [2].

Contrairement aux heuristiques, les méthodes d'approximation assurent une information de garantie sur la qualité des solutions trouvées par rapport à l'optimum global. Les heuristiques peuvent être aussi divisées en deux sous-classes: les heuristiques dédiées qui sont des méthodes spécifiques à un problème donné, et les métaheuristiques qui regroupent des méthodes génériques non dédiées à un problème particulier [2]. La notion d'heuristique date de 1945 [8] et les premières métaheuristiques, dont les algorithmes évolutionnaires et les premières recherches locales, datent des années 60. Depuis cette époque, les métaheuristiques ont rencontré de plus en plus de succès et ont vu leur nombre augmenter (plus d'une dizaine de grandes familles d'algorithmes). Cela est dû à leur capacité à résoudre des problèmes de taille importante en des temps raisonnables en apportant des solutions de qualité satisfaisante. Il est bien sûr toujours possible d'obtenir des solutions de meilleures qualités mais cela ne peut se faire qu'au détriment du coût de la méthode utilisée. Deux familles de métaheuristiques peuvent être distinguées : les métaheuristiques à base de solution unique et celles à base de population de solutions. Les premières consistent en une amélioration d'une seule solution existante tant que cela est possible, ce sont donc des méthodes avec un fort pouvoir d'intensification. Par contre, les métaheuristiques à base de population permettent de mieux appréhender les grands espaces de recherche de part leur pouvoir de diversification. Même si les métaheuristiques sont applicables à n'importe quel type de problème, il convient toujours d'analyser les caractéristiques du problème à traiter pour évaluer la meilleure méthode à utiliser [9] [10].

I.5 Les métaheuristiques à base de population de solutions

Les métaheuristiques à population de solutions sont des méthodes stochastiques qui font évoluer, au fur et à mesure des itérations, une population de solutions dans l'espace de recherche, dans le but d'obtenir de meilleures solutions à un problème d'optimisation. L'utilisation de plusieurs solutions simultanément permet naturellement de renforcer la diversité de la recherche et d'augmenter la possibilité d'émergence de solutions de bonne qualité. On distingue deux familles de métaheuristiques à base de population : les algorithmes évolutionnaires et les algorithmes d'intelligence en essaim. Une grande variété de méthodes

basées sur une population de solutions a été proposée dans la littérature dont certaines d'entre-elles sont décrites dans la suite.

I.5.1 Les algorithmes évolutionnaires

Les algorithmes évolutionnaires «evolutionary algorithms (EAs)» forment une branche d'algorithmes inspirés de l'évolution biologique des espèces, élaborés au cours des années 1950 [11]. L'idée principale de cette dernière repose sur le fait que les individus qui ont hérité des caractères bien adaptés à leur milieu ont tendance à vivre assez longtemps pour se reproduire, tandis que les faibles ont tendance à disparaître. Dans ce sens, de nombreuses méthodes ont été développées dont les algorithmes génétiques [12], l'évolution différentielle [13] [14], les stratégies d'évolution [15] et la programmation évolutive [16].

I.5.1.1 Algorithme à évolution différentielle

L'algorithme à évolution différentielle «differential evolution (DE)» est une stratégie de recherche stochastique à base de population. L'algorithme DE a été introduit par Storn et Price en 1995 [13]. Il est fondé sur les mêmes opérateurs que les algorithmes génétiques, c'est-à-dire la mutation, le croisement et la sélection. La distinction principale en construisant de meilleures solutions est que l'algorithme DE se base sur l'opération de mutation alors que les algorithmes génétiques se fondent sur le croisement. En effet, la technique d'optimisation DE utilise la différence de deux vecteurs sélectionnés de façon aléatoire afin de générer un nouveau vecteur de solutions. L'algorithme utilise l'opération de mutation comme un mécanisme de recherche et l'opérateur de sélection pour diriger la convergence vers les régions potentielles dans l'espace de recherche. L'algorithme DE utilise également un croisement non uniforme qui peut prendre des paramètres de vecteur d'enfant d'un seul parent. Cette série de transformation permet d'évoluer la population d'une manière progressive jusqu'à l'obtention de solutions satisfaisantes.

La population initiale est générée aléatoirement, considérons une population de N individus. Chaque individu est un vecteur de D-dimensions, représenté par $x_i^k = (\, x_{1i}^k,\ x_{2i}^k,\ \ldots,\ x_{Di}^k)$ avec $i = 1,2,\ldots,N$ et k désigne la génération. Les opérateurs de mutation, croisement et de sélection sont ensuite appliqués à chaque génération pour produire un vecteur d'essai $u_i^{k+1} = (\, u_{1i}^{k+1},\ u_{2i}^{k+1},\ldots,\ u_{Di}^{k+1})$ avec $i = 1,2,\ldots,N$.

a) Mutation : pour chaque vecteur x_i^k, un vecteur mutant est produit en utilisant l'une des stratégies de mutation suivantes :

$$Rand/1 : v_i^{k+1} = x_{r1}^k + F \cdot \left(x_{r2}^k - x_{r3}^k\right) \quad \text{(I.1)}$$

$$Best/1 : v_i^{k+1} = x_{best}^k + F \cdot \left(x_{r1}^k - x_{r2}^k\right) \quad \text{(I.2)}$$

$$Current - to - best/1 : v_i^{k+1} = x_i^k + F \cdot \left(x_{r1}^k - x_{r2}^k\right) + F \cdot \left(x_{best}^k - x_i^k\right) \quad \text{(I.3)}$$

$$Best/2 : v_i^{k+1} = x_{best}^k + F \cdot \left(x_{r1}^k - x_{r2}^k\right) + F \cdot \left(x_{r3}^k - x_{r4}^k\right) \quad \text{(I.4)}$$

$$Rand/2 : v_i^{k+1} = x_{r1}^k + F \cdot \left(x_{r2}^k - x_{r3}^k\right) + F \cdot \left(x_{r4}^k - x_{r5}^k\right) \quad \text{(I.5)}$$

Les indices $r1, r2, r3, r4\ et\ r5 \in \{1, 2, \ldots, N\}$ sont mutuellement différents et distincts de l'indice i. $F \in [0, 2]$ est le coefficient de mutation qui contrôle l'amplitude des mutations. x_k^{best} est le meilleur individu à la génération k.

b) Croisement : le vecteur d'essai final u_i^{k+1} est formé en passant par le croisement qui combine le vecteur parent x_i^k et son vecteur mutant v_i^{k+1}.

$$u_{ij}^{k+1} = \begin{cases} v_{ij}^{k+1} & si \quad (rand(j) \leq CR \quad ou \quad j = j_{rand}) \\ x_{ij}^k & ailleurs \end{cases} \quad \text{(I.6)}$$

où $rand(j)$ est la $j^{ème}$ valeur fournie par un générateur de nombre aléatoire uniforme appartenant à l'intervalle [0,1]. CR est le coefficient de croisement qui appartient à l'intervalle [0,1] et est déterminé par l'utilisateur. j_{rand} est un indice choisi au hasard dans l'ensemble $\{1, 2, \ldots, N\}$.

c) Sélection : l'opérateur de sélection compare les deux vecteurs de solutions u_i^{k+1} et x_i^k, pour tirer le meilleur individu qu'il doit ensuite choisir dans la prochaine génération. Les principales étapes de l'algorithme DE sont décrites dans l'Algorithme I.1.

Algorithme I.1 : Pseudo-code d'optimisation à évolution différentielle

Initialiser la population aléatoirement;
Evaluer la fitness $f(x_i)$ de chaque individu;
Répéter
  Pour tous les individus i **Faire**
    Choisir aléatoirement $r1, r2, r3, r4\ et\ r5$ avec $r1 \neq r2 \neq r3 \neq r4 \neq r5$;
    Créer un vecteur mutant v_i ;
    Créer le vecteur d'essai u_i à partir du mutant v_i et de x_i ;
    Si $f(u_i) < f(x_i)$ **Alors** remplacer x_i par u_i ;
  Fin Pour
Jusqu'à atteindre le critère d'arrêt.

I.5.1.2 Algorithme génétique

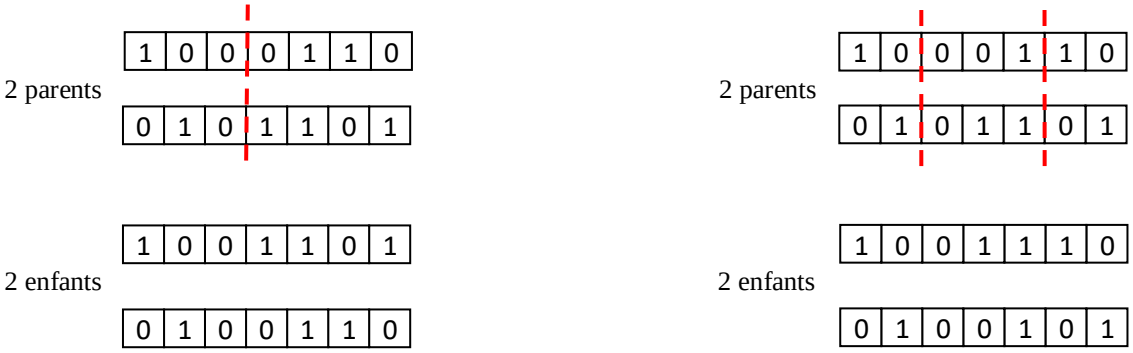
Les algorithmes génétiques développés par John Holland [12], constituent la technique d'optimisation la plus célèbre dans les calculs évolutionnaires. L'algorithme génétique «genetic algorithm (GA)» est une métaheuristique stochastique à base de population qui se fond sur des opérateurs inspirés de l'évolution des espèces et de la sélection naturelle.

Un algorithme génétique est formé d'une population de solutions appelées individus, chaque individu est représenté par un chromosome qui est analogue au chromosome des systèmes biologiques. La première étape du processus de recherche de l'algorithme est de générer aléatoirement une population initiale de chromosomes. Chaque chromosome représente le codage d'une solution potentielle qui dépend principalement du problème à traiter, le codage binaire est largement utilisé grâce à sa simplicité [17]. De génération en génération, les individus évoluent à l'aide des opérateurs génétiques inspirés de l'évolution naturelle qui sont la sélection, le croisement, la mutation et le remplacement jusqu'à ce que la condition d'arrêt soit remplie. L'Algorithme I.2 retrace les étapes principales de l'algorithme GA.

a) Sélection : la sélection est un opérateur qui détermine à partir de la génération précédente quels individus de la population qui assuraient la reproduction de la génération suivante. En effet, le mécanisme de sélection définit le degré selon lequel les individus de meilleure qualité, suivant leurs valeurs de la fonction d'évaluation, ont plus de chance d'être sélectionnés. Ce degré de sélection est lié au taux de convergence de l'algorithme GA, plus les degrés de sélection sont élevés, plus les taux de convergence sont élevés aussi. Il existe plusieurs stratégies de sélection. Les stratégies les plus connues sont la sélection par roulette, la sélection par tournois et la sélection par rang [18].

b) Croisement (recombinaison) : l'idée générale du croisement est l'échange d'information entre les individus parents de manière à générer de nouveaux individus (enfants) qui héritent partiellement des caractéristiques des parents. Il existe de très nombreuses variantes de l'opérateur de croisement dans la littérature, les plus simples sont le croisement à un point et le croisement à deux points (Figure I.2).

c) Mutation : l'opérateur de mutation consiste en la modification aléatoire et occasionnelle de faible probabilité de la valeur d'un composant de l'individu (Figure I.3). La mutation permet de conserver la diversité de la population pour enrichir l'exploration de tout l'espace de recherche et d'assurer une recherche aussi bien globale que locale afin d'éviter la convergence prématurée de l'algorithme GA [18].



a) croisement à un point b) croisement à deux points

Figure I.2 : Illustration de l'opérateur de croisement.

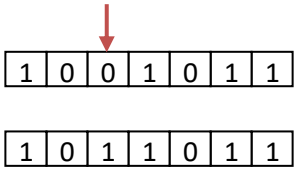


Figure I.3 : Illustration de l'opérateur de mutation.

Algorithme I.2 : Pseudo-code d'un algorithme génétique

Initialiser la population de N_{AG} chromosomes aléatoirement;
Evaluer la fitness $f(x_i)$ de chaque individu (chromosome);
Nouvelle population créer une nouvelle population en répétant les étapes suivantes jusqu'à l'accomplissement de la nouvelle population :
  Sélection sélectionner deux chromosomes parents à partir de la population en fonction de leur fitness (la meilleure fitness a une grande chance d'être sélectionnée);
  Croisement Croiser les parents pour former des enfants;
  Mutation avec une probabilité de mutation, muter les nouveaux individus (enfants);
  Remplacement utiliser la nouvelle population générée pour une future génération de l'algorithme;
  Tester si le critère d'arrêt est atteint, stopper;
Boucle revenir à l'étape d'évaluation de fitness

d) Remplacement : l'opérateur de remplacement a pour but de garder les meilleurs individus de la génération précédente, constituée de parents et d'enfants qui représenteront les parents de la génération suivante.

I.5.2 L'intelligence en essaim

L'intelligence en essaim «swarm intelligence (SI)» est une discipline de l'intelligence artificielle qui regroupe une variété d'algorithmes inspirés des comportements collectifs de certaines espèces, telles que les colonies de fourmis, les volées d'oiseaux, les bancs de poissons, et les colonies d'abeilles. Ces algorithmes s'appuient généralement sur des mécanismes comportementaux qui sont propres à ces espèces, à l'exemple du processus de fourragement chez les abeilles, la recherche de nourriture chez les fourmis et bien d'autres. L'idée de base de l'intelligence en essaim est de transformer les capacités collectives caractérisant le comportement naturel des insectes en techniques artificielles adaptées à la résolution de problèmes d'optimisation difficile. L'intelligence en essaim est née de la modélisation mathématique et informatique des phénomènes biologiques rencontrés en éthologie [19]. Bien que les capacités individuelles puissent être simples et limitées (par exemple, une fourmi se contente de suivre la trace laissée par une autre), elles permettent à la collectivité de résoudre des problèmes difficiles, tels que la recherche du chemin le plus court entre le nid et une source de nourriture, parmi d'innombrables voies possibles. Plusieurs métaheuristiques à base de l'intelligence en essaim ont été introduites telles que l'optimisation par colonies de fourmis [20], l'optimisation par essaims de particules [21], l'optimisation par colonies d'abeilles artificielles [22] et l'algorithme de chauves-souris [23].

I.5.2.1 Optimisation par colonies de fourmis

L'optimisation par colonies de fourmis «Ant Colony Optimization (ACO)» est une métaheuristique basée sur l'intelligence par essaim, introduite par Colorni, Dorigo et Maniezzo [20] au début des années 1990 pour résoudre les problèmes d'optimisation difficile. Elle résulte de l'observation des fourmis réelles au cours de leur exploration des sources de nourriture. Malgré leurs capacités physiques et cognitives limitées, les fourmis peuvent réaliser des taches très complexes comme la recherche de nourriture par coopération en regroupant leurs capacités disponibles et leurs expériences collectives. Les fourmis vivent en colonies et elles ont collectivement la capacité de trouver le plus court chemin entre leur nid et une source de nourriture via une substance chimique volatile, appelée phéromone. Cette substance est un moyen de communication entre les fourmis qui travaillent en parfaite collaboration. La figure I.4 illustre le comportement d'une colonie de fourmis pour s'approvisionner. Un obstacle est placé sur le trajet des fourmis qui, après une étape d'exploration, finiront par emprunter le plus court chemin entre le nid et la source de

nourriture [24]. En se déplaçant du nid à la source de nourriture et vice-versa, toutes les fourmis déposent leurs phéromones sur leur trajet. Les fourmis qui sont retournées le plus rapidement au nid en passant par la source de nourriture sont celles qui ont emprunté le chemin le plus court. Ainsi, la densité de phéromones déposées par unité de temps sur ce trajet est légèrement plus élevée que celle présente sur le trajet le plus long, ce qui permet d'augmenter le nombre de fourmis qui suivent cette trajectoire. De plus, la quantité de la phéromone s'évapore au fil du temps, cette action appelée processus d'évaporation, d'où le chemin le plus long est de moins en moins emprunté et sa trace disparaîtra presque complètement (Figure I.4d).

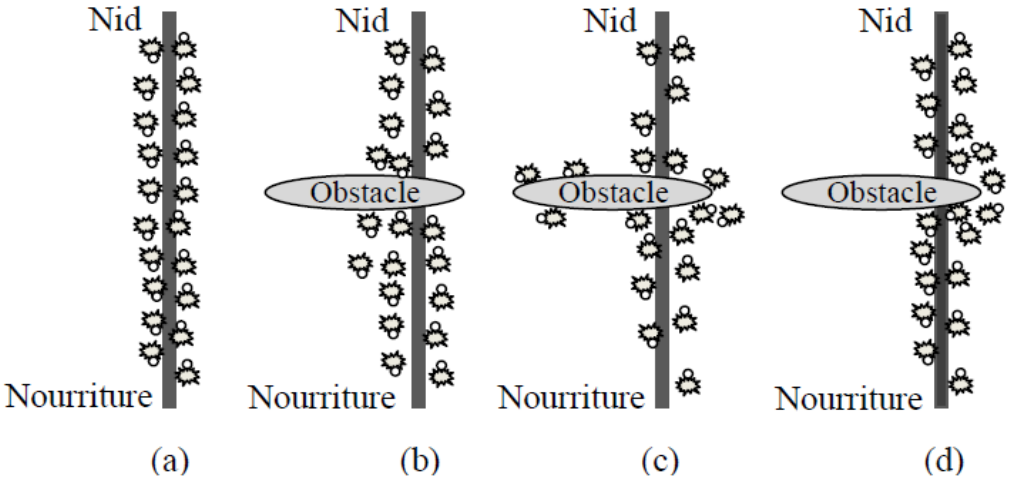


Figure I.4 : Illustration de la capacité des fourmis à chercher de la nourriture en minimisant leur parcours. (a) Recherche sans obstacle, (b) Introduction d'un obstacle, (c) Recherche du chemin optimal, (d) Chemin optimal trouvé.

L'optimisation par colonies de fourmis a été conçue initialement pour résoudre le problème de voyageur de commerce puis elle a considérablement évolué pour résoudre d'autres problèmes d'optimisation. Ce problème consiste à trouver le chemin le plus court reliant n villes données, chaque ville ne devant être visitée qu'une seule fois. L'étape initiale de l'algorithme de colonies de fourmis est de placer aléatoirement chaque fourmi dans une ville, puis se déplacer au fil des itérations d'une ville à une autre construisant ainsi un trajet. À chaque itération t, chaque fourmi k choisit de se déplacer d'une ville i où elle se trouve vers une ville j pas encore visitée avec une probabilité donnée par :

$$p_{ij}^{k}(t) = \begin{cases} \frac{(\tau_{ij}(t))^{\alpha}.(\eta_{ij})^{\beta}}{\sum_{l\in J_i^k}(\tau_{il}(t))^{\alpha}.(\eta_{ij})^{\beta}}, & si\ \ j \in J_i^k \\ 0, & si\ \ j \notin J_i^k \end{cases} \quad \text{(I.7)}$$

où α et β sont deux paramètres qui contrôlent l'importance relative de l'intensité et la visibilité. $\tau_{ij}(t)$ est la densité de trace de phéromone déposée sur l'arc qui relie les deux villes i et j à l'itération t. $\eta_{ij} = \frac{1}{d_{ij}}$ est une mesure de visibilité, utilisée pour diriger le choix des fourmis vers les villes les plus proches et ainsi, éviter la répétition des villes. J_i^k est le voisinage faisable de la fourmi k.

Après une exécution complète, chaque fourmi dépose une certaine quantité de phéromone $\Delta\tau_{ij}^{k}$ sur l'ensemble de son parcours. Cette quantité est déterminée par :

$$\Delta\tau_{ij}^{k}(t) = \begin{cases} \frac{Q}{L^k(t)}, & si\ \ (i,j) \in T^k(t) \\ 0, & si\ (i,j) \notin T^k(t) \end{cases} \quad \text{(I.8)}$$

où $L^k(t)$ est la longueur du trajet parcouru par la fourmi k. Q est une constante. $T^k(t)$ est le chemin traversé par la fourmi k durant l'itération t.

Un processus d'évaporation est introduit afin d'éviter que toutes les fourmis ne convergent prématurément vers les bonnes solutions et maximiser la diversification dans l'espace de recherche. La règle de mise à jour des traces de phéromones est la suivante :

$$\tau_{ij}(t+1) = (1-\rho).\tau_{ij}(t) + \sum_{k=1}^{m}\Delta\tau_{ij}^{k}(t) \quad \text{(I.9)}$$

où ρ est un paramètre d'évaporation de phéromone, m est le nombre de fourmis.

Le pseudo code de l'algorithme de colonies de fourmis pour le problème du voyageur de commerce peut être résumé par l'Algorithme I.3.

Algorithme I.3 : Pseudo-code d'optimisation par colonies de fourmis

Tant que le critère d'arrêt n'est pas satisfait **Faire**
  Pour chaque fourmi k **Faire**
    Choisir une ville au hasard;
    Pour chaque ville non visitée i **Faire**
      Choisir une ville j, dans la liste J_i^k des villes restantes selon (I.7);
    Fin Pour
    Déposer une piste $\Delta\tau_{ij}^{k}(t)$ sur le trajet $T^k(t)$ conformément à (I.8);
  Fin Pour
Évaporer les pistes selon (I.9)

I.5.2.2 Optimisation par essaims de particules

L'optimisation par essaim de particules «Particle Swarm Optimization (PSO)» est une métaheuristique à population de solutions proposée par Kennedy et Eberhart en 1995 [21]. L'idée originale est fondée sur le comportement social des animaux évoluant en essaim dans la nature, telle que les poissons et les oiseaux. En effet, on peut observer chez ces animaux des dynamiques de déplacement relativement complexes, alors qu'individuellement chaque individu a une intelligence limitée et une connaissance locale de sa situation dans l'essaim. Un individu de l'essaim ne dispose que d'une connaissance de la position et la vitesse de ses plus proches voisins. Chaque individu utilise cette information locale sur ses plus proches voisins, ainsi que sa propre mémoire, pour décider de son propre déplacement. Des règles simples, telles que «aller dans la même direction» ou «rester proche de ses voisins», suffisent à maintenir la cohésion de l'essaim, et permettent la mise en œuvre de comportements collectifs complexes et adaptatifs. L'optimisation par PSO met en jeu un ensemble d'individus appelés "particules" qui font partie d'une population nommée "essaim", initialement générée de façon aléatoire dans l'espace de recherche. Chaque particule représente une solution potentielle au problème d'optimisation et se déplace dans l'espace de recherche, en quête de l'optimum global. Ces particules forment des chemins repérés par des vecteurs de position quasi-aléatoires. Dans un espace de recherche de dimension D, chaque particule i est caractérisée par son vecteur position $x_i = (x_{i1}, x_{i2}, \dots, x_{iD})$ et son vecteur vitesse $v_i = (v_{i1}, v_{i2}, \dots, v_{iD})$. Durant le processus, chaque particule est attirée vers sa propre position locale $pbest = (pbest_{i1}, pbest_{i2}, \dots, pbest_{iD})$ et la meilleure position atteinte par toutes les particules de l'essaim $gbest = (gbest_1, gbest_2, \dots, gbest_D)$. Le vecteur de vitesse et le vecteur de position sont calculés à partir de l'équation (I.10) et de l'équation (I.11), respectivement, pour chaque itération k :

$$v_{ij}^{k+1} = wv_{ij}^{k} + c1r1_{ij}^{k}\left[pbest_{ij}^{k} - x_{ij}^{k}\right] + c2r2_{ij}^{k}\left[gbest_{j}^{k} - x_{ij}^{k}\right] \quad \text{(I.10)}$$

$$x_{ij}^{k+1} = x_{ij}^{k} + v_{ij}^{k+1}, \quad j \in \{1, 2, \dots, D\},\ i = 1,2, \dots, N \quad \text{(I.11)}$$

où w désigne le coefficient d'inertie, les coefficients $c1$ et $c2$ sont des constantes, appelées coefficients d'accélération, $r1$ et $r2$ sont deux nombres aléatoires qui suivent une distribution uniforme sur $[0, 1]$ et N le nombre de particules de l'essaim. L'Algorithme I.4 représente le pseudo-code de la méthode PSO.

Afin d'améliorer les performances de l'algorithme PSO, de nombreuses variantes ont été rapportées dans la littérature. Un état de l'art sur l'algorithme PSO et ses variantes est présenté dans [25].

Algorithme I.4 : Pseudo-code d'optimisation par essaim de particules

Initialiser aléatoirement les positions et les vitesses de chaque particule;
Evaluer la fitness $f(x_i)$ de chaque particule;
Répéter
  Pour toutes les particules i **Faire**
    Calculer les vitesses des particules selon l'équation (I.10);
    Calculer les positions des particules selon l'équation (I.11);
    Calculer $f(x_i)$ de chaque particule;
    Si $f(x_i) < f(pbest_i)$ **Alors** $pbest_i = x_i$;
    Si $f(x_i) < f(gbest_i)$ **Alors** $gbest_i = x_i$;
  Fin Pour
Jusqu'à atteindre le critère d'arrêt.

I.5.2.3 Optimisation par colonies d'abeilles

Dans la nature, les abeilles manifestent des comportements très complexes en rapport avec l'accouplement, la reproduction et la recherche de nourriture. Une panoplie d'algorithmes d'optimisation sont inspirés de ces comportements naturels qui se produisent en ou à partir de colonies. Parmi les algorithmes inspirés du comportement d'accouplement et de reproduction chez les abeilles, on peut citer l'algorithme MBO (*Marriage in Honey Bees Optimization*) [26]. L'algorithme MBO commence par une seule reine sans famille et passe au développement d'une colonie avec une famille ayant une ou plusieurs reines. Dans la littérature, on recense plusieurs versions de l'approche MBO, telle que l'algorithme HBMO (*Honey-Bees Mating Optimization*) [27], l'algorithme FMHBO (*Fast Marriage in Honey Bees Optimization*) [28] et l'algorithme HBO (*Honey-Bees Optimization*) [29].

Une autre classe d'algorithmes à essaim d'abeilles est adaptée au comportement naturel de fourragement chez les abeilles. Ces algorithmes utilisent une recherche évolutive standard ou exploratoire aléatoire pour localiser des emplacements prometteurs. Ils utilisent également la recherche exploitante sur les sites les plus prometteurs pour trouver l'optimum global. Les algorithmes suivants ont été inspirés du comportement de recherche de nourriture chez les abeilles : l'algorithme BS (*Bee System*) [30], l'algorithme BCO (*Bee Colony Optimization*) [31], l'algorithme ABC *(Artificial Bee Colony*) [32] et l'algorithme BA (*Bees Algorithm*) [33].

L'algorithme BS est une version améliorée de l'algorithme génétique (GA). Il s'y inspire de manière à améliorer la recherche locale tout en préservant la capacité de recherche globale de l'algorithme GA.

L'algorithme BCO a été proposé pour résoudre des problèmes d'optimisation combinatoire. Il repose sur deux phases appelées passe-avant et passe-arrière. Une solution partielle est générée lors de l'étape de passage-avant avec une exploration individuelle et une expérience collective, qui sera ensuite utilisée à l'étape passe-arrière. Dans la phase de passage-arrière, l'information de probabilité est utilisée pour prendre la décision de continuer à explorer la solution actuelle dans la prochaine passe-avant ou de commencer le voisinage des nouvelles solutions sélectionnées. La nouvelle solution est déterminée en utilisant des techniques probabilistes telles que la sélection de la roulette (*Roulette Wheel Selection)*.

L'optimisation par colonies d'abeilles artificielles (*Artificial Bee Colony (ABC))* a été proposée par Karaboga en 2005 [32]. L'algorithme ABC se base sur les groupes d'abeilles suivants : les abeilles *Employed*, les abeilles *Onlooker* et les abeilles *Scout*. Les abeilles *Employed* exploitent et retournent à la ruche avec des informations sur le paysage. Cette information de recherche est partagée avec les abeilles *Onlooker*, qui à leur tour évaluent cette information avec une approche probabiliste (par exemple la méthode de la roulette [22]) pour lancer une recherche dans le voisinage. Pendant ce temps, les abeilles *Scout* effectuent une recherche aléatoire pour effectuer l'exploration. Le chapitre II sera exclusivement dédiée à cette approche d'optimisation.

L'algorithme BA emploie lui aussi des processus de recherche locale et globale. La procédure est différente par rapport à l'algorithme ABC qui fait appel à un mécanisme probabiliste dans la recherche de voisinage; l'algorithme BA quant à lui emploie une procédure se basant sur l'évolution de la fitness pour conduire la recherche [34].

I.6 Applications de l'optimisation en ingénierie des systèmes

Les métaheuristiques, y compris les algorithmes évolutionnaires et les algorithmes d'intelligence en essaim, ont montré leur performance sur une large variété de problèmes. En effet, la simplicité et la robustesse de ces méthodes ont suscité l'intérêt de plusieurs communautés de chercheurs qui ont mené des études d'application de plusieurs problèmes dans différents domaines du monde réel [35] [36], et en particulier les problèmes d'ingénierie [37]. Les métaheuristiques sont facilement adaptables à tout type de problème d'optimisation. Un certain nombre de travaux ont porté sur l'optimisation de fonctions mathématiques complexes, comme celles présentées dans [38] [39]. Un certain nombre d'applications concernent la robotique pour la recherche de trajectoire optimale [40], le contrôle de processus [41] [42], le traitement d'images [43], le dimensionnement des circuits électriques [44], le transport pour le problème de tournées de véhicules [45], les télécommunications pour

le problème de la conception de réseaux radio [46], etc. On recense aussi des applications en intelligence artificielle traitant des problèmes particuliers comme les réseaux de neurones artificiels [47], les systèmes d'inférence floue [48], les problèmes de classification non supervisée (le clustering des données) [49], les systèmes d'inférence adaptatifs neuro-floues [50], etc. Les métaheuristiques sont également appliquées aux problèmes de l'identification des systèmes, l'optimisation des réseaux électriques, la planification des tâches, la classification des données, l'exploration de données, la conception technique, l'optimisation dans les systèmes photovoltaïques et bien d'autres domaines.

I.7 Conclusion

Ce chapitre est consacré à l'état de l'art des outils et concepts de l'optimisation. Nous y avons présenté les principales définitions, la formulation générale du problème d'optimisation ainsi que les différentes méthodes de résolution.

Il en découle de cette description succincte qu'il n'existe aucune méthode d'optimisation qui est capable de résoudre efficacement tous les problèmes d'optimisation difficile. C'est l'essence même du théorème du "No Free Lunch" qui conclut sur la supériorité d'un algorithme en l'associant à l'aptitude de la méthode à résoudre une classe donnée de problèmes. L'expérience de l'utilisateur se présente comme seul guide dans le choix de l'algorithme à implanter.

En outre, le succès d'une méthode est lié à l'équilibre entre deux objectifs contradictoires, à savoir l'exploration et l'exploitation. L'exploration est liée à la capacité de l'algorithme à prospecter les différentes régions inconnues dans l'espace de recherche pour fournir une estimation fiable de l'optimum global. Tandis que, l'exploitation se réfère à la capacité d'utiliser les connaissances existantes (les meilleures solutions trouvées) pour déterminer l'optimum global. Une meilleure exploration permet de préserver la diversité de la population et explorer de nouvelles régions afin d'éviter la convergence prématurée vers des optimums locaux. Une meilleure exploitation permet, entre autre, d'accélérer la convergence et fournir des solutions de bonne qualité. Lors de toute conception, l'effort se concentre autour du maintien de l'équilibre entre l'exploration et l'exploitation; chose qui s'analyse souvent empiriquement à travers l'expérience. C'est dans cette perspective que l'on projette notre contribution à l'optimisation par essaims d'abeilles dans ce travail de thèse.

Chapitre II

Optimisation par colonies d'abeilles artificielles

II.1 Introduction

Les abeilles sont des insectes sociaux qui vivent en colonies, elles sont très organisées et très rigoureuses. Dans une colonie d'abeilles, les individus sont organisés pour effectuer les différentes tâches nécessaires à la survie de la colonie. En effet, ils sont engagés dans plusieurs activités: la recherche de nourriture, l'accouplement, la défense de la ruche, l'essaimage, etc. D'une manière générale, la colonie d'abeilles caractérise un système où le travail collectif des entités interagissant entre-elles, fait émerger un comportement complexe global. La coopération entre les individus est réalisée d'une manière distribuée sans aucun mécanisme de contrôle centralisé, ce qui rend les modèles de l'intelligence en essaim assez manipulables.

De nombreux algorithmes tentent d'utiliser les principes du comportement collectif des abeilles pour résoudre plusieurs problèmes difficiles dans différents domaines. Généralement, on distingue deux principales classes d'algorithmes d'optimisation qui se fondent sur le comportement des abeilles :

- Les algorithmes qui s'inspirent des mécanismes génétiques et du comportement naturel associés au processus d'accouplement d'abeilles.
- Les algorithmes inspirés du comportement naturel associé à la recherche de nourriture chez les abeilles.

Les algorithmes de la première classe se fondent sur l'idée qu'une colonie d'abeilles comprend un grand nombre d'individus qui sont génétiquement hétérogènes en raison de l'accouplement multiples de la reine. Les algorithmes basés sur la recherche de nourriture chez les abeilles utilisent généralement des abeilles artificielles pour rechercher des solutions caractérisées comme étant des sources de nourriture. Selon le nombre et la qualité des sources de nourriture (solutions) trouvées, un sous-ensemble de la population d'abeilles artificielles explore l'environnement (espace de recherche) pour découvrir de nouvelles sources de nourriture (création de nouvelles solutions), tandis que les abeilles restantes exploitent l'environnement autour des sources de nourriture explorées afin d'essayer de trouver de

meilleures solutions en effectuant des opérations de recherche locales dans le but d'améliorer une solution existante [51].

Il existe de nombreux algorithmes d'optimisation inspirés du comportement des abeilles, l'algorithme de colonies d'abeilles artificielles [22] est celui qui attire de plus en plus les chercheurs [52]. C'est une méthode stochastique relativement récente, sa popularité est liée d'une part à la facilité de sa mise en œuvre et, d'autre part, à ses capacités de résolution vérifiées sur des problèmes d'optimisation multidimensionnels [53].

Le présent chapitre introduit d'abord quelques concepts fondamentaux liés au monde des abeilles biologiques, suivi d'une description du modèle comportemental des abeilles lors de la recherche de nourriture. En s'appuyant sur ces notions et concepts, on présentera dans la suite la structure algorithmique de base de l'optimisation par colonies d'abeilles, ainsi que certaines variantes qui ont été développées dans la littérature concernée. Les applications de cette procédure d'optimisation, les problématiques de conception et de mise en œuvre y seront également passées en revue.

II.2 L'essaim d'abeilles

Les abeilles sont des insectes sociaux qui vivent en colonie représentée par une ruche. Elles sont hautement développées et participent ainsi à diverses tâches complexes. Dans une colonie d'abeilles on trouve trois types d'abeilles: une reine, des faux-bourdons et des ouvrières. La reine est la seule abeille femelle fertile dans la ruche et son seul rôle est d'assurer la survie de la colonie en donnant naissance à des abeilles dans la ruche. Les faux-bourdons sont des abeilles mâles, dont le seul rôle est de féconder la reine lors de son vol d'accouplement. Les ouvrières sont des femelles sexuellement non développées et possèdent des organes biologiques tels que les glandes alimentaires, les glandes odoriférantes, les glandes de cire et les paniers à pollen qu'elles impliquent dans les différentes tâches journalières comme le fait de nettoyer la ruche, nourrir la couvée, prendre soin de la reine, garder la ruche, manipuler le nectar entrant et chercher de la nourriture (nectar, pollen, etc.) [52].

Le comportement intelligent de l'essaim d'abeilles repose sur les deux concepts fondamentaux suivants : le concept d'auto-organisation et le concept de répartition des tâches [22]. L'auto-organisation dans un essaim d'abeilles peut être définie comme un ensemble de mécanismes dynamiques conduisant à la construction des colonies (modèles, structures) à partir d'un grand nombre d'interactions entre ses différentes composantes. Ces mécanismes établissent des règles de base qui régissent le comportement collectif des abeilles de la même

colonie. Les règles établies font en sorte que les interactions sont exécutées sur la base des informations recueillies localement (non complexes), sans aucun rapport avec le modèle complexe global de l'essaim. À l'intérieur d'un essaim, différentes tâches sont exécutées simultanément par des agents (abeilles) spécialisés. La répartition des tâches au sein de la colonie trouve son efficacité dans le traitement distribué de ces tâches, qui est loin d'être nécessairement séquentiel. La performance collective de l'essaim d'abeilles est exclusivement basée sur ces deux concepts.

II.3 Modèle comportemental de recherche de nourriture des essaims d'abeilles

II.3.1 Caractérisation du modèle

La recherche de nourriture est l'une des tâches complexes réalisées par un essaim d'abeilles, le modèle minimal de la sélection de fourrage qui conduit à l'émergence de l'intelligence collective des essaims d'abeilles se compose de trois éléments essentiels: les sources de nourriture, les abeilles employées dites «Employed» et les abeilles non employées dites «Onlooker» et «Scout». Le modèle définit aussi deux modes principaux du comportement: le recrutement à une source de nourriture et l'abandon d'une source.

a) Sources de nourriture

La qualité d'une source de nourriture dépend de nombreux facteurs tels que sa proximité de la ruche, sa richesse, sa concentration d'énergie, et la facilité d'extraction de cette énergie. Par souci de simplicité, la rentabilité d'une source de nourriture peut être évaluée à partir d'un seul descripteur appelé fonction d'adéquation.

b) Abeilles employées

Elles sont associées à une source de nourriture particulière en qualité d'exploitants effectifs du site ou bien employées pour l'exploitation. Elles diffusent des informations sur cette source particulière, sa distance, sa direction et sa rentabilité et partagent ces informations avec une certaine probabilité.

c) Abeilles non employées

Elles se trouvent continuellement en quête d'une source de nourriture à exploiter. On distingue deux types possibles d'abeilles non employées: les abeilles dites "Scout" qui cherchent de nouvelles sources en prospectant le voisinage de la ruche et les abeilles "Onlooker" qui attendent dans la ruche pour recevoir les informations partagées par les abeilles employées. Le nombre d'abeilles Scout est environ 5-10% de la population.

II.3.2 Mécanisme de communication

L'échange d'informations entre les différentes catégories d'abeilles est l'évènement le plus important dans la formation de la connaissance collective. Cet échange qui vise à partager des éléments clefs relatifs à la source de nourriture à exploiter s'effectue à travers des mécanismes naturels propres aux abeilles. Le phénomène particulier identifié sous le nom de «waggle dance» est l'un des mécanismes de communication qui permet l'échange d'informations entre les abeilles.

Dès que les informations sur toutes les sources sont disponibles aux abeilles Onlooker à travers l'analyse des différentes danses présentées par les abeilles Employed, la décision concernant le recrutement du site le plus rentable est rapidement prise. Afin de mieux comprendre le comportement de fourragement chez les essaims d'abeilles, nous proposons d'examiner le schéma illustré en Figure II.1.

Considérons les deux sources de nourritures A et B. Au début, une abeille de la colonie dépourvue de tâches spécifiques peut être appelée à accomplir l'un des rôles suivants :

- Elle peut être une abeille Scout et commence à prospecter spontanément le voisinage de la ruche pour rechercher une source de nourriture possible. Ce comportement peut être à l'origine de certaines motivations internes ou externes (les flèches orientées S sur la figure II.1).
- Elle peut être appelée à exploiter une source de nourriture après avoir reçue les informations correspondantes (les flèches R sur la figure II.1).

Après avoir localisé la source de nourriture, l'abeille utilise sa propre capacité pour mémoriser son emplacement et passe immédiatement à la phase d'exploitation. Ainsi, l'abeille deviendra une abeille "Employed". L'abeille Employed prélève alors une charge de nectar depuis la source et retourne à la ruche afin de la décharger. Après déchargement, l'abeille pourra se manifester selon l'un des trois cas de figures suivantes :

– Elle n'aura aucune tâche spécifique après l'abandon de la source de nourriture (UF).

– Elle partagera les informations sur la source découverte avec d'autres abeilles de la colonie et reviendra à l'exploitation de la même source (UF1).

– Elle continue à exploiter la source qu'elle a découverte sans transférer les informations aux autres abeilles (UF2).

II.4 L'algorithme de colonies d'abeilles artificielles

L'optimisation par colonies d'abeilles artificielles «Artificial Bee Colony (ABC) Optimization» est une métaheuristique basée sur l'intelligence en essaim qui s'inspire du

modèle comportemental des abeilles à la recherche de nourriture. C'est une procédure relativement récente qui a été proposée en 2005 par le Professeur Karaboga [22].

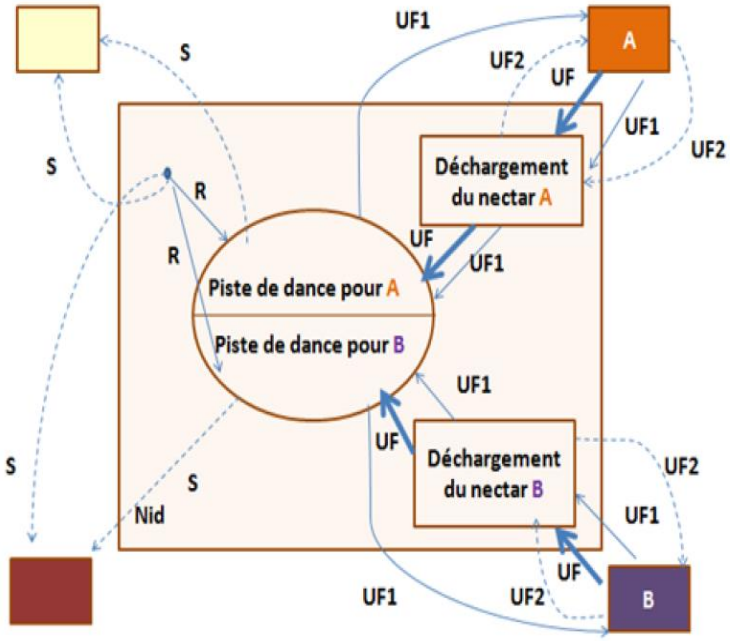


Figure II.1 : Schéma illustratif du comportement d'une abeille lors du fourragement [22].

Comme dans le modèle minimal du fourragement d'un essaim d'abeilles, l'algorithme de colonies d'abeilles artificielles (ABC) utilise une population d'individus composée de trois catégories d'abeilles: les abeilles "Employed", les abeilles "Onlooker" et les abeilles "Scout". La colonie est équitablement répartie entre abeilles Employed et abeilles Onlooker. Chaque abeille est affectée à une source de nourriture particulière. En d'autres termes, le nombre des abeilles Employed est égal au nombre des abeilles Onlooker alors que ce nombre est égal aussi au nombre des sources de nourriture. L'abeille Employed dont la source de nourriture est épuisée devient une abeille Scout. L'algorithme II.1 résume les principales étapes de l'optimisation par ABC.

Chaque cycle de recherche se compose de trois étapes : la phase des abeilles Employed, la phase des abeilles Onlooker et la phase des abeilles Scout. Dans la structure de base de l'algorithme ABC, la position d'une source de nourriture donnée représente une solution potentielle au problème d'optimisation et la qualité de la solution correspond à la qualité de nectar de la position associée. Les abeilles Onlooker sont affectées aux sources de nourriture à base d'un mécanisme de sélection probabiliste dite "gourmande". La probabilité de sélectionner une source de nourriture par une abeille Onlooker est proportionnelle à la quantité de nectar, si la teneur en nectar d'une source est élevée, la probabilité d'être

sélectionnée augmenterait. Les colonies d'abeilles abritent également les abeilles Scout qui sont en fait identifiées comme étant les explorateurs de la colonie. Les abeilles Scout n'ont pas d'orientation particulière, elles se basent sur l'exploration du voisinage de la ruche pour rechercher des sources éventuelles.

Algorithme II.1 : Structure générale de l'algorithme ABC

1. Initialisation
2. **Répéter**
   3. Phase des abeilles Employed : Envoyer les abeilles Employed aux sources de nourriture et mettre à jour leurs positions.
   4. Phase des abeilles Onlooker : Sélectionner les sources riches en nectar à base d'une fonction d'adéquation et mettre à jour les solutions.
   5. Phase des abeilles Scout : Repérer la solution la plus inactive (épuisement de site), et la remplacer par une autre générée aléatoirement.
6. **Jusqu'à** atteindre le critère d'arrêt

Dans l'algorithme ABC, si une solution (source de nourriture) représentée par une abeille Employed n'est pas améliorée (la source de nourriture est épuisée) après un nombre prédéterminé d'essais ou d'itérations, elle sera donc abandonnée par l'abeille Employed puis remplacée, et l'abeille Employed deviendra par la suite une abeille Scout. Le nombre d'essais nécessaire à l'abandon d'une source est égal à la valeur du seuil *"limit"* qui représente un paramètre de contrôle important de l'algorithme ABC.

II.4.1 Modèle de base de l'algorithme ABC

Comme tous les algorithmes à base de population de solutions, l'algorithme ABC utilise une population qui est constituée d'un nombre de solutions possibles représentées par les positions des sources de nourriture dont la teneur en nectar correspond à la qualité de la solution. Les phases de l'algorithme ABC peuvent être expliquées comme suit:

Initialisation de la population : l'algorithme commence par l'initialisation d'une population de SN solutions $(x_1, x_2, \dots, x_{SN})$, générées aléatoirement dans l'espace de recherche. Chaque solution $x_i = \{x_{i1},\ x_{i2}, \dots, x_{iD}\}$, où D représente la dimension du problème, est initialisée en utilisant l'équation de recherche suivante:

$$x_i^j = x_{min}^j + rand(0,1)\left(x_{max}^j - x_{min}^j\right) \quad \text{(II.1)}$$

où $i = 1, 2, \dots, SN$, $j = 1, 2, \dots, D$, x_{min}^j et x_{max}^j sont les limites inférieures et supérieures de la dimension j.

Ensuite, pour un problème de minimisation, chaque solution est évaluée par l'équation (II.2) :

$$fit_i(x_i) = \begin{cases} \frac{1}{f_i(x_i)+1} & f_i(x_i) \geq 0 \\ 1 + |f_i(x_i)| & f_i(x_i) < 0 \end{cases} \quad \text{(II.2)}$$

où $f_i(x_i)$ est la fonction objectif évaluée pour la solution x_i, $fit_i(x_i)$ est la fonction d'adéquation (fitness).

Pour chaque solution, on initialise à zéro un compteur de tentatives d'amélioration de solutions T_i, $i = 1, 2, \dots, SN$.

Après l'initialisation, la population de solutions est soumise à des cycles répétés jusqu'à atteindre le critère d'arrêt de la recherche. Ces cycles représentent des processus de recherches réalisés par les abeilles Employed, les abeilles Onlooker et les abeilles Scout.

La phase d'abeilles «Employed» : chaque abeille Employed tente à améliorer sa position en déterminant une nouvelle solution potentielle v_i en utilisant l'équation :

$$v_i^j = x_i^j + \phi_i^j \left(x_i^j - x_k^j\right) \quad \text{(II.3)}$$

où $i = 1, 2, \dots, SN$, $j \in \{1, 2, \dots, D\}$ et $k \in \{1, 2, \dots, SN\}$ sont des indices choisis aléatoirement tels que $k \neq i$, x_k est une source voisine. ϕ_i^j est un nombre aléatoire dans l'intervalle $[-1, 1]$. Une fois la position v_i obtenue, elle sera évaluée à partir de l'équation (II.2) et comparée à x_i. Un processus de sélection dite "gourmande" est appliqué: lorsque la valeur de la fonction fitness calculée selon l'équation (II.2) pour la nouvelle solution est supérieure à celle de la précédente, l'abeille mémorise la nouvelle position et abandonne l'ancienne. Le compteur T_i sera alors mis à zéro. Dans le cas inverse, elle conserve l'ancienne solution, le compteur T_i sera alors incrémenté.

La phase d'abeilles «Onlooker» : À la fin du processus de recherche, les abeilles Employed partagent toutes les informations sur les positions et leurs teneurs en nectar avec les abeilles Onlooker en manifestant une danse particulière dite "waggle dance". Les abeilles Onlooker choisissent une source de nourriture en fonction de la probabilité p_i associée à cette source, et qui est définie par :

$$p_i = \frac{fit_i}{\sum_{n=1}^{SN} fit_n} \quad \text{(II.4)}$$

Les abeilles Onlooker utilisent aussi l'équation (II.3) pour améliorer les solutions choisies, l'équation (II.2) pour évaluer les solutions courantes et le processus de sélection *"gourmande"* pour les comparer. Si une solution est améliorée, le compteur T_i correspondant sera mis à zéro. Dans le cas contraire, le compteur sera incrémenté.

La phase d'abeilles «Scout» : lorsque la valeur du compteur T_i d'une abeille Employed ou d'une abeille Onlooker dépasse un certain seuil du paramètre de contrôle "limit", l'abeille Employed décidera alors d'abandonner sa solution actuelle et deviendra une abeille Scout qui aura pour rôle de déterminer une nouvelle solution aléatoirement par l'équation (II.1). À chaque cycle de recherche, une seule abeille Scout est utilisée au plus. Le pseudo-code de l'algorithme ABC est donné par l'algorithme II.2, l'organigramme associé est représenté dans la figure II.2.

Algorithme II.2 : Pseudo-code d'optimisation par colonies d'abeilles artificielles

1. Générer une population initiale de SN solutions par l'équation (II.1)
2. Evaluer la population initiale par l'équation (II.2)
3. Cycle=1
4. **Répéter**
   5. Pour chaque abeille Employed:
      6. Produire une nouvelle solution v_i par l'équation (II.3)
      7. Calculer la fonction fitness de la nouvelle solution par l'équation (II.2)
      8. Appliquer le processus de la sélection gourmande
      9. Calculer la probabilité p_i de chaque solution x_i par l'équation (II.4)
   10. Pour chaque abeille Onlooker:
      11. Choisir une solution x_i en fonction de sa probabilité p_i
      12. Produire une nouvelle solution v_i par l'équation (II.3)
      13. Calculer la fonction fitness de la nouvelle solution par l'équation (II.2)
      14. Appliquer le processus de sélection gourmande
   15. Vérifier s'il existe une solution abandonnée:
      16. Alors, remplacer cette solution par une autre générée aléatoirement par l'équation (II.1)
   17. Mémoriser la meilleure solution trouvée
   18. Cycle=Cycle+1
19. **Jusqu'à** atteindre le critère d'arrêt

II.4.2 Les variantes de l'algorithme ABC

L'algorithme ABC est initialement conçu pour l'optimisation dans le domaine continu [22], son efficacité et ses performances ont été comparées à d'autres algorithmes qui sont très connus et largement utilisés dans le domaine d'optimisation difficile [32][53-54]. Les résultats obtenus ont fait l'œuvre de nombreux travaux, proposant de nombreuses variantes visant à améliorer l'algorithme ABC. En fait, malgré l'efficacité et la simplicité de la mise en œuvre de l'algorithme d'optimisation par colonies d'abeilles artificielles, ce dernier et comme toute autre métaheuristique souffre du problème d'équilibrage entre exploration et exploitation.

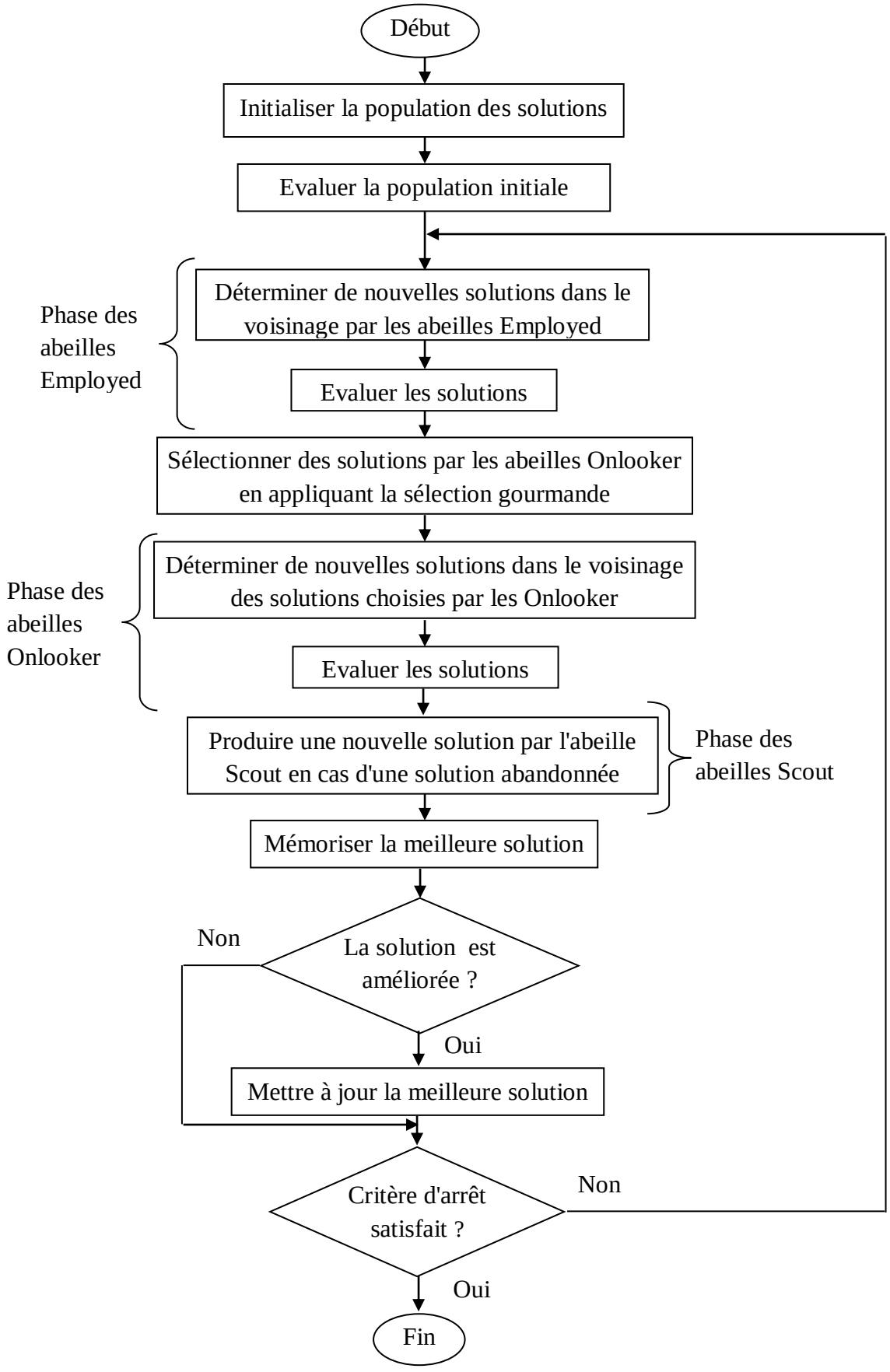


Figure II.2 : Organigramme de l'optimisation par colonies d'abeilles artificielles.

Les variantes de l'algorithme ABC peuvent être catégorisées en trois groupes, selon la description suivante :

- **Variantes basées sur la modification de l'équation de recherche de solutions :**

Afin d'améliorer l'exploitation de l'algorithme ABC, Zhu et al. [55] s'inspirent de l'algorithme PSO en incluant le meilleur individu dans l'équation de recherche de solutions. Une pondération de l'information de la meilleure solution trouvée jusqu'alors est intégrée dans l'équation de recherche de l'algorithme ABC de base. Dans la variante proposée, l'équation suivante est utilisée à la place de l'équation (II.3).

$$v_{i,j} = x_{i,j} + \phi_{i,j}\left(x_{i,j} - x_{k,i}\right) + \psi_{i,j}\left(x_{best,j} - x_{i,j}\right), \quad i \neq k, \tag{II.5}$$

où $\psi_{i,j}$ est un nombre aléatoire dans $[0, C]$, C est un constant.

Une autre variante qui se repose sur la meilleure solution est proposée dans [56]. L'équation de recherche définie pour les abeilles Onlooker est modifiée comme suit :

$$v_{i,j} = x_{i,j} + \phi_{i,j} fit_b\left(x_{i,j} - x_{best,j}\right) \tag{II.6}$$

où fit_b est la valeur de fitness de la meilleure solution. Une autre modification au niveau des abeilles Scout basée sur l'utilisation de l'équation suivante a été également introduite :

$$v_{i,j} = x_{i,j} + \phi_{i,j}\left[\omega_{max} - \frac{iteration}{MCN}(\omega_{max} - \omega_{min})\right] x_{i,j}, \tag{II.7}$$

où ω_{max} et ω_{min} sont des coefficients d'ajustement. De plus, le processus de la sélection gourmande ne se base plus sur la valeur de fitness mais sur l'évaluation par la fonction objectif de la solution.

- **Variantes basées sur l'hybridation avec d'autres approches d'optimisation :**

Le travail de Kumar et al. [57] présente une hybridation de l'algorithme ABC avec l'algorithme GA, les auteurs intègrent l'opérateur de croisement de l'algorithme GA dans le modèle ABC de base pour améliorer les capacités d'exploration et d'exploitation. Une hybridation de l'algorithme ABC et de l'algorithme d'évolution quantique (QEA) est proposée dans [58] pour résoudre les problèmes d'optimisation continus. L'algorithme ABC est adopté pour renforcer la capacité de la recherche locale et augmenter le caractère aléatoire de la population. Les stratégies de croisement et de mutation du QEA sont intégrées dans la phase des abeilles Employed pour produire de nouvelles solutions.

- **Variantes basées sur l'introduction de nouveaux paramètres de contrôle :**

L'algorithme ABC améliore ses solutions en modifiant une seule dimension $j \in \{1, 2, \ldots, D\}$, pour accélérer la convergence. Akay et al. proposent dans [59] un nouveau paramètre de contrôle MR qui permet de modifier plus qu'une dimension à la fois. L'équation de recherche est donnée par:

$$v_{i,j} = \begin{cases} x_{i,j} + \phi_{i,j}(x_{i,j} - x_{k,j}), & si\ \ R_{i,j} < MR \\ x_{i,j} \quad , & sinon \end{cases} \tag{II.8}$$

Pour chaque dimension $j = 1, 2, \ldots, D$, la valeur de $x_{i,j}$ sera modifiée si un nombre choisi aléatoirement $R_{i,j} \in [0, 1]$ est inférieur à MR. De plus, dans l'algorithme ABC de base $\phi_{i,j}$ varie dans l'intervalle $[-1, 1]$, tandis que la variante proposée introduit un pas adaptatif de déplacement SF $(scaling\ factor)$, ce qui permet de tolérer une variation dans l'intervalle $[-SF, SF]$. L'ajustement automatique du facteur SF est effectué de manière cyclique selon (II.9) :

$$SF(t+1) = \begin{cases} SF(t) * 0.85 & , \ si\ \varphi(m) < 1/5, \\ SF(t)/0.85 & , \ si\ \varphi(m) > 1/5, \\ SF(t) & , \ si\ \ \varphi(m) = 1/5, \end{cases} \tag{II.9}$$

où $\varphi(m)$ est le ratio des déplacements améliorants. Les valeurs de ces coefficients dépendent du type du problème appréhendé. Les meilleurs résultats sont obtenus pour $MR = 0.4$ et $SF = 1$.

Au regard des performances démontrées sur des problèmes continus mono-objectif, le modèle de base de l'algorithme ABC a été également adapté aux problèmes discrets [60], aux problèmes avec contraintes [61], aux problèmes multi-objectif [62], et bien d'autres. Pour surmonter le problème de temps de calcul, plusieurs versions parallèles de l'algorithme ABC sont présentées dans la littérature [63].

II.5 Applications de l'algorithme ABC

L'algorithme de colonies d'abeilles artificielles a fait l'objet d'une multitude d'applications dans des domaines variés tels que l'électronique, le traitement d'image, la bioinformatique, l'exploration de données, et bien d'autres.

Dans [64], Karaboga et al. ont utilisé l'algorithme ABC comme outil d'apprentissage appliqué aux réseaux de neurones artificiels à travers l'ajustement des poids synaptiques. L'étude expérimentale réalisée sur trois problèmes benchmark de classification a clairement démontré la performance de l'algorithme ABC, comparée au modèle d'optimisation par rétro-programmation et aux algorithmes génétiques.

En 2012, Akay et al. ont proposé l'emploi de l'algorithme ABC pour la résolution de problèmes d'optimisation de grande dimension [65]. Des tests sur neuf fonctions numériques benchmark sans contraintes, et cinq autres problèmes d'optimisation avec contraintes sont considérés dans l'étude pour illustrer les performances de la méthode ABC. Les résultats comparatifs appuient visiblement la stratégie ABC.

Dans [66], les auteurs ont examiné l'application du modèle ABC en classification floue en considérant une approche hybride pour la détection de tumeurs à partir d'images IRM.

Dans le domaine de contrôle de procédés, Gozde et al. [67] ont étudié le problème d'ajustement des paramètres du contrôleur PID en utilisant le modèle ABC. Les résultats comparatifs au modèle PSO ont permis de dégager les performances de l'algorithme ABC.

Pour les problèmes discrets, Yin et al. [68] ont pu avancer un mécanisme d'adaptation de l'algorithme ABC pour résoudre les problèmes d'optimisation combinatoire. L'algorithme proposé est utilisé pour résoudre le problème combinatoire d'ordonnancement Job-Shop. L'étude expérimentale est réalisée sur des benchmarks du problème. Les résultats obtenus ont prouvé encore une fois la performance de la version discrète de l'algorithme ABC proposée.

Dans le cadre de l'optimisation multi-objectif, Prajati et Srivastava [69] ont présenté une application de l'algorithme ABC au problème d'optimisation de la puissance réactive dans les réseaux électriques. L'efficacité de l'approche proposée est validée sur un réseau test standard de type IEEE 30-Bus.

Comme il peut être constaté, les applications de la stratégie ABC touchent pratiquement à plusieurs domaines d'intérêt, une raison pour laquelle l'appréhension de cette métaheuristique demeure une piste de recherche prometteuse. Les références [70] [71] passent en revue autant d'autres applications.

II.6 Problématique et contraintes

II.6.1 Cadre d'évaluation

L'existence d'une grande variété de problèmes d'optimisation rend la comparaison des différents algorithmes d'optimisation et le choix d'une méthode pour un problème donné une tache plutôt difficile. Selon le théorème "no free lunch" [72], aucun algorithme d'optimisation n'est plus performant que les autres pour tous types de problèmes. Une comparaison entre deux algorithmes d'optimisation sur un ensemble de problèmes possibles peut conduire des performances similaires en moyenne. Par conséquence, la performance d'une méthode donnée doit être testée selon le type du problème appréhendé.

Une collection de problèmes d'optimisation continus présentée sous forme de fonctions de référence génériques, a été mise en place pour évaluer les performances des algorithmes d'optimisation. Les performances escomptées sont généralement explicitées en termes de vitesse de convergence, précision et robustesse. Les fonctions benchmark ont des

caractéristiques que l'on retrouve dans des problèmes réels difficiles. Elles peuvent être catégorisées en trois parties [73] :

- **Les fonctions unimodales :** ce sont des fonctions considérées comme faciles. Elles n'ont qu'un seul optimum local (qui est donc l'optimum global). Une simple méthode d'optimisation locale permet de les résoudre.
- **Les fonctions faiblement multimodales :** ces fonctions possèdent quelques optimums locaux, mais en densité relativement faible par rapport à la taille de l'espace de recherche. Elles sont plus difficiles à résoudre que les premières, car il faudrait explorer suffisamment l'espace de recherche pour trouver le bon optimum.
- **Les fonctions hautement multimodales :** ces fonctions sont souvent les plus difficiles à résoudre, car elles contiennent un grand nombre d'optimums locaux. Il est alors difficile de converger simplement vers l'optimum global.
- **Les fonctions discontinues :** une fonction qui présente des «sauts» est discontinue. La notion de saut correspond à l'existence d'une limite à droite et d'une limite à gauche qui ne valent pas la même chose.

La séparabilité est une autre caractéristique importante des fonctions benchmarks. On parle de fonction séparable si l'on peut la minimiser (ou la maximiser) en minimisant (maximisant) ses composantes prises séparément. Les fonctions non séparables sont donc plus difficiles à optimiser, car leurs variables peuvent être interdépendantes, ce qui implique la non-séparabilité de ses composantes [73].

De part les caractéristiques énumérées ci-dessus, des fonctions benchmarks typiques sont décrites dans la suite.

a) Les fonctions benchmarks standards

Elles sont au nombre de 22 fonctions ($F1 - F22$) telles que reporté dans [74]. Ces fonctions sont sélectionnées avec des propriétés différentes: unimodales, multimodales, séparables et non séparables pour une évaluation de performance. L'annexe A reprendra les principales fonctions standards utilisées le long de cette thèse.

b) Les fonctions benchmarks de type CEC

Les fonctions CEC dodelinent les topologies de l'espace de recherche. En effet, il a été montré que certaines particularités topologiques de fonctions objectifs peuvent favoriser certains algorithmes. Par exemple, il existe des algorithmes qui ont la capacité de converger plus rapidement si l'optimum global est situé à l'origine $[0,0,\ldots,0]$ de l'espace de recherche ou au centre de celui-ci. Pour éviter cela, les fonctions définies dans la base CEC sont translatées

par rapport à leurs formes d'origine et/ou tournées. Certaines fonctions présentent aussi la particularité d'avoir leur optimum global situé sur une des frontières de l'espace de recherche [75].

Les benchmarks CEC2005 sont définies dans le cadre de la conférence "*2005 IEEE Congress on Evolutionary Computation* (CEC'05)" [76]. La base CEC2005 est composée de 25 fonctions ($f1 - f25$) qui peuvent être classées en quatre groupes: les fonctions $f1$ à $f5$ sont unimodales, et les fonctions $f6$ à $f25$ sont multimodales. Une discontinuité est appliquée à la fonction $f23$ et une rotation est appliquée à plusieurs autres fonctions.

D'autre part, les benchmarks CEC2013 sont définies dans le cadre de la conférence "*2013 IEEE Congress on Evolutionary Computation* (CEC'13)" [77], avec une base de 28 fonctions ($f1 - f28$). Les fonctions $f1$ à $f5$ sont unimodales, les fonctions $f6$ à $f20$ sont multimodales et les fonctions $f21$ à $f28$ sont des fonctions composées. Une discontinuité est appliquée à la fonction $f13$ et une rotation est appliquée aux autres fonctions, à l'exception des fonctions $f1$, $f5$, $f11$, $f14$, $f17$, $f19$ et $f22$.

La description des fonctions CEC'05 et CEC'13 est fournie en Annexe A.

II.6.2 Critère d'évaluation

Afin d'évaluer la performance des algorithmes, plusieurs critères sont considérés [73], dont les principaux sont :

- **La qualité de la solution :** Au lieu d'utiliser le temps d'exécution, qui est dépendant de la machine sur laquelle l'algorithme est lancé, on calcule le nombre d'évaluations de la fonction objectif nécessaire pour atteindre une précision donnée. On peut inversement fixer le nombre d'évaluations de la fonction objectif et calculer l'erreur obtenue. De plus, pour éviter d'obtenir un résultat biaisé, on lance l'algorithme plusieurs fois sur la même fonction objectif et on calcule une moyenne et un écart-type des erreurs. Il s'agit d'une analyse statistique d'une approche stochastique qu'on est appelé à mener pour une étude fondée de performance.
- **L'invariance :** Il s'agit de vérifie la robustesse de l'algorithme par rapport aux translations ou à l'augmentation de la dimension pour une même fonction objectif.

II.6.3 Critère d'arrêt

Le critère d'arrêt est la condition qui mène à un arrêt programmé de l'algorithme. Généralement, il joue un rôle très important dans le jugement de la qualité des résultats. Le critère d'arrêt peut être différent suivant le problème posé. Si l'optimum global est connu a priori, on peut définir une valeur cible de la fonction objectif comme critère d'arrêt. Sinon, il

est commun de considérer comme critère d'arrêt l'absence d'amélioration de la fonction objectif sur plusieurs itérations consécutives ou fixer un nombre maximum d'évaluations ou d'itérations.

II.6.4 Mise en œuvre et limitations

L'optimisation par colonies d'abeilles est relativement une "récente" métaheuristique, si elle est comparée à bien d'autres qui datent depuis des décennies. Elle a pris beaucoup d'intérêt depuis son apparition, ce qui a conduit plusieurs chercheurs à s'atteler sur cette problématique d'optimisation en considérant des domaines variés de l'optimisation difficile. Sur le plan implémentation, le modèle ABC présente beaucoup d'avantages liés à la simplicité de mise en œuvre et d'hybridation avec d'autres métaheuristiques, et au faible nombre de paramètres de contrôle qu'il utilise.

Malgré les performances tant prouvées dans plusieurs travaux, les problèmes de convergence dans certains cas d'étude (fonctions unimodales) et de convergence prématurée dans d'autres cas (fonctions multimodales) restent cependant posés [78]. Ce sont des contraintes qui sont essentiellement liées aux aspects d'exploration et d'exploitation qui constituent l'objet de toute éventuelle amélioration de performance. De plus, et comme toutes autres métaheuristiques, son efficacité pourrait être altérée en fonction de la dimension du problème traité. Une première piste de travail identifié dans ce sens a trait au mécanisme de recherche qu'adopte le modèle ABC de base et qui se fond sur l'utilisation d'une seule règle de recherche à base de voisinage unique, effectué sur une seule dimension à chaque mise à jour de solutions. C'est à partir de cette constatation fondamentale qu'on s'est défini un cadre de contribution à l'optimisation par colonies d'abeilles, dont la démarche de conception et de validation est décrite dans le Chapitre III.

II.7 Conclusion

Dans ce chapitre, nous nous sommes particulièrement intéressés à l'optimisation par colonies d'abeilles artificielles (ABC), objet du présent travail de thèse. Pour mieux appréhender cette métaheuristique relevant du domaine de l'intelligence en essaim, nous avions d'abord présenté les principaux concepts de la méthode qui s'inspirent fondamentalement du comportement biologique des essaims d'abeilles lors du fourragement. De ce fait, il a été d'abord question de décrire le modèle comportemental y afférent, avant d'expliciter le mécanisme de recherche de l'optimisation par essaims d'abeilles. Le modèle

algorithmique de base de l'approche ABC est ensuite détaillé, suivie d'une présentation succincte de certaines de ses variantes et ses applications.

Les problématiques de conception, d'évaluation et de mise en œuvre du concept ABC y sont également retracées, dont les conclusions ont permis de nous définir un cadre bien ciblé de contribution à l'optimisation par essaims d'abeilles. Sur ce plan précis, il en ressort des contraintes et des limitations qui touchent essentiellement au mécanisme de recherche globale sur lequel se fond le modèle ABC, qui relève certaines insuffisances ayant trait à l'exploration et l'exploitation, la recherche globale unidimensionnelle et la structure algorithmique simplifiée. Ces considérations sont à la base de l'étude de développement de la nouvelle approche par essaims d'abeilles proposée qui fera l'objet du Chapitre III.

Chapitre III

Nouvelle approche d'optimisation par essaims d'abeilles à apprentissage coopératif

III.1 Introduction

Le comportement biologique des abeilles lors du fourragement a autant constitué une source d'inspiration pour les chercheurs qui s'attèlent à calquer des mécanismes comportementaux en vue de développer de nouvelles stratégies d'optimisation à essaim d'abeilles. D'après [71], le modèle ABC étant le plus répandu avec un nombre d'applications assez conséquent couvrant différentes classes de problèmes. Le modèle ABC s'est montré jusque-là imposant par rapport à d'autres métaheuristiques, en raison d'abord de la simplicité de sa structure algorithmique et puis au faible nombre de paramètres de contrôle qu'il utilise. Cette particularité a contribué à la conception d'une panoplie de variantes à performances distinctes [70] [79].

Le modèle ABC n'échappe cependant pas à certaines insuffisances d'ordre computationnel, qui sont l'objet de plusieurs études récentes. Le compromis entre exploitation et exploration est éventuellement commun à toutes les métaheuristiques; il représente en fait une des contraintes auxquelles est confrontée toute étude de développement d'approches d'optimisation à recherche stochastique, et ce vu l'impact qu'il a sur la convergence de l'algorithme notamment pour les problèmes de grande dimension. Ainsi, notons qu'un grand nombre de variantes du modèle ABC de base sont proposées au sens de l'amélioration des capacités d'exploitation et d'exploration.

Notre contribution s'inscrit justement au cœur de cette problématique. La nouvelle approche d'optimisation que nous proposons se résume à deux nouvelles variantes du modèle ABC auquel des modifications structurelles et paramétriques sont opérées par l'intégration de modèles algorithmiques modélisant des mécanismes spécifiques caractérisant les essaims d'abeilles. Le présent chapitre retrace les détails de la contribution de la thèse et analyse les performances et la robustesse de la stratégie d'optimisation proposée sur des problèmes d'optimisation numériques complexes (difficiles).

III.2 La stratégie d'optimisation proposée

Comme souligné au chapitre II, l'efficacité d'un processus de recherche est étroitement liée à l'équilibre entre exploration et exploitation: une meilleure exploration préserverait la diversité de la population afin d'éviter la convergence prématurée, alors qu'une meilleure exploitation permettrait d'accélérer la convergence et d'améliorer la qualité des solutions. L'algorithme d'optimisation par colonies d'abeilles artificielles (ABC) utilise les abeilles Scout pour contrôler le processus d'exploration et les abeilles Employed et Onlooker pour le contrôle du processus d'exploitation [32]. En effet, les abeilles Employed et Onlooker utilisent la même procédure de recherche décrite par l'équation (II.3), explicitée dans le chapitre II, et qui se fond sur le concept de voisinage pour mettre à jour les positions des sources de nourriture. Le concept de voisinage employé dans la version de base du modèle ABC apporte une très grande souplesse à la structure algorithmique de la métaheuristique, mais en revanche limite sa capacité d'exploitation [55]. Les travaux visant à améliorer cette grandeur importante ne manquent cependant pas, dont certains font appel à des modèles de recherche hybrides ou encore des équations de recherche multiples [80] [81].

La nouvelle stratégie d'optimisation par essaims d'abeilles à apprentissage coopératif proposée dans le cadre de ce travail et dénommée CLABC (Cooperative Learning Artificial Bee Colony) [82] se veut une version modifiée du modèle ABC de base. Les modifications apportées ne se limitent pas à un simple paramétrage ou ajustement de paramètres de contrôle, mais touchent essentiellement à la structure algorithmique de l'algorithme ABC par l'introduction de concepts comportementaux propres aux essaims d'abeilles, modélisés moyennant des mécanismes appropriés.

D'un point de vue conceptuel, les concepts comportementaux que nous avions convenus d'introduire reposent sur la description linguistique suivante :

"*Dans un essaim d'abeilles, les abeilles Employed et les abeilles Onlooker peuvent être divisées en sous-groupes d'abeilles. Les sous-groupes de fourrageuses Employed et Onlooker pourraient effectuer des fourragements de manière indépendante, engager à cet effet des outils de recherche spécifiques à chaque niveau et échanger mutuellement les informations recueillies lors du processus de recherche de nourriture* ".

La projection d'une telle description comportementale sur une plate-forme algorithmique nous conduit d'abord à faire ressortir les processus suivants en vue de les modéliser :

- Le processus de partitionnement des essaims d'abeilles qui est à la base du concept d'apprentissage coopératif.

- Le processus de recherche multiple basé sur l'association de différentes règles d'exploitation.
- Le processus d'altération des caractéristiques des sources de nourriture.

L'appréhension de cette problématique nous a permis dans une première phase d'établir une première variante de l'approche proposée que nous avions dénommé par CLABC (Cooperative Learning Artificial Bee Colony) dont les résultats sont publiés dans [83]. Le modèle CLABC emploie un mode de recherche multiple associé aux fourrageuses Employed responsables de la phase d'exploitation. Une variante modifiée de l'approche CLABC, dénommée mCLABC (modified CLABC) [82], est ensuite développée. Les modifications apportées au modèle CLABC interviennent à différents niveaux, tant structurel que paramétrique. Dans la suite, une description détaillée des deux modèles proposés est fournie.

III.2.1 L'algorithme CLABC

III.2.1.1 Structure et mécanisme de recherche

Le modèle CLABC représente la première variante de la stratégie d'optimisation proposée. Il se fond sur l'emploie d'un mécanisme de recherche à règles multiples intégrées de différentes manières dans les phases Employed et Onlooker. Les abeilles Employed sont divisées en 3 groupes; chaque groupe forme une sous-population de solutions potentielles et utilise des équations de recherche de solutions différentes de celles définies pour les autres groupes d'abeilles. Les abeilles Onlooker sont catégorisées en un seul groupe. Le mécanisme de recherche du modèle CLABC est décrit par l'ensemble des règles suivantes :

$$v_i^j = x_i^j + \varphi_i^j\left(x_i^j - x_k^j\right),\ i = 1,2,\dots,SN,\ i \neq k,\ j \in \{1,2,\dots,D\} \quad \text{(III.1)}$$

$$v_i^j = x_k^j + \varphi_i^j\left(x_i^j - x_{k1}^j\right),\ i = 1,2,\dots,SN,\ i \neq k \neq k_1,\ j \in \{1,2,\dots,D\} \quad \text{(III.2)}$$

$$v_i^j = x_i^j + r_i^j\left(x_i^j - x_{mean}^j\right) + r1_i^j\left(x_k^j - x_{k1}^j\right),\ i = 1,2,\dots,SN, i \neq k \neq k_1, j \in \{1,2,\dots,D\} \quad \text{(III.3)}$$

$$v_i^j = x_{best}^j + r_i^j\left(x_i^j - x_{mean}^j\right) + r1_i^j\left(x_k^j - x_{k1}^j\right),\ i = 1,2,\dots,SN, i \neq k \neq k_1, j \in \{1,2,\dots,D\} \quad \text{(III.4)}$$

$$v_i^j = x_k^j + \varphi_i^j\left(x_{k1}^j - x_{k2}^j\right),\ i = 1,2,\dots,SN,\ i \neq k \neq k_1 \neq k_2,\ j \in \{1,2,\dots,D\} \quad \text{(III.5)}$$

$$v_i^j = x_{best}^j + \varphi_i^j\left(x_k^j - x_{k1}^j\right),\ i = 1,2,\dots,SN,\ i \neq k \neq k_1,\ j \in \{1,2,\dots,D\} \quad \text{(III.6)}$$

où $\varphi_i^j \in [-1,1]$, $r_i^j, r1_i^j \in [0,1]$ sont des nombres aléatoires générés pour chaque dimension à chaque instant de traitement, x_i^j, x_{k1}^j et x_{k2}^j représentent les $j^{ième}$ dimensions des solutions choisies aléatoirement parmi les sous-populations, les indices k, k_1, k_2 sont générés aléatoirement dans l'intervalle $[1, SN]$ et doivent être mutuellement différents de l'indice

courant i, x_{best}^{j} désigne la $j^{ième}$ dimension de la meilleure solution globale obtenue jusque-là par les sous-populations, et x_{mean}^{j} est la solution moyenne correspondant à la $j^{ième}$ dimension de toutes les solutions dans les sous-populations.

III.2.1.2 Procédure de recherche

Dans la phase Employed, les équations de recherche (III.1) et (III.2) sont appelées par le premier groupe d'abeilles Employed pour générer de nouvelles solutions candidates. Une seule équation de recherche est utilisée pour effectuer la mise à jour de la solution en fonction d'une probabilité croissante définie par $(iter/MCN)^{1/3}$, où $iter$ est le nombre de génération courant et MCN est le nombre maximal de générations. Plus précisément, si une valeur aléatoire choisie entre 0 et 1 est inférieure ou égale à la probabilité prédéfinie, alors l'équation (III.1) serait utilisée, sinon, le mécanisme de recherche se réalise selon l'équation (III.2). Sur la base de la même définition de probabilité donnée ci-dessus, les équations (III.3) et (III.4) sont utilisées pour le deuxième groupe et les équations (III.5) et (III.6) pour le dernier.

Le caractère unidimensionnel de l'algorithme ABC de base induit un faible taux de convergence. De ce fait, et pour augmenter la vitesse de convergence de l'algorithme, nous proposons d'introduire un procédé d'altération sélectif de dimensions de sorte que les abeilles Employed puissent commencer la recherche en changeant une seule dimension choisie aléatoirement au même titre que le modèle ABC de base. Ensuite, lorsque le nombre d'itérations atteint la valeur $iter = MCN/4$, l'algorithme optimiserait de manière aléatoire un seul paramètre ou tous les paramètres simultanément. Vers la fin du processus itératif, quand le nombre d'itérations est entre $iter = MCN/2$ et $iter = MCN$, les abeilles Employed font évoluer leurs solutions en changeant toutes les dimensions.

Par la suite, après l'achèvement de la phase Employed et compte tenu du processus de *sélection gourmande*, les abeilles Onlooker commencent leur recherche à partir d'un déplacement unidimensionnel en utilisant l'équation (III.2). Ensuite, lorsque $iter = MCN/4$, les abeilles Onlooker procèdent à la mise à jour aléatoire de toutes les dimensions de la solution en utilisant l'équation (III.1) ou bien une seule dimension choisie aléatoirement à l'aide de l'équation (III.2). L'algorithme III.1 décrit la stratégie d'optimisation basée sur le modèle CLABC.

III.2.2 L'algorithme mCLABC

Le modèle mCLABC est la deuxième variante de la stratégie d'optimisation proposée. Il s'agit d'une version modifiée du modèle CLABC incarnant davantage le caractère coopératif

des essaims d'abeilles et les mécanismes de recherche et de sélection multiples par l'introduction de modifications structurelles et paramétriques dans le modèle CLABC.

Algorithme III.1 : Pseudo-code de l'algorithme CLABC

1. Initialisation
2. Générer une population initiale de **SN** solutions par l'équation (II.1)
3. Evaluer la population initiale par l'équation (II.2)
4. $\boldsymbol{Cycle = 1, FEs = SN}$
5. **Répéter**
6. **La phase des abeilles Employed:**
7. $\boldsymbol{C = (Cycle/MaxCycle)\text{^}(1/3)}$
8. **Pour** $\boldsymbol{i = 1}$ à $\boldsymbol{SN/3}$
9. **Si** $(\boldsymbol{rand(0,1) <= C})$
10. Produire une nouvelle solution $\mathbf{v_i}$ par l'équation (III.1) en changeant une ou toutes les dimensions
11. **Sinon**
12. Produire une nouvelle solution $\mathbf{v_i}$ par l'équation (III.2) en changeant une ou toutes les dimensions
13. **Fin Si**
14. Evaluer $\mathbf{f(v_i)}$ et mettre $\boldsymbol{FEs = FEs + 1}$
15. Appliquer le processus de la sélection gourmande
16. **Fin Pour**
17. **Pour** $\boldsymbol{i = (SN/3) + 1}$ à $\boldsymbol{(2 * SN/3)}$
18. Répéter les étapes 9-15 en utilisant les équations (III.3) et (III.4)
19. **Fin Pour**
20. **Pour** $\boldsymbol{i = (2 * SN/3) + 1}$ à $\boldsymbol{SN}$
21. Répéter les étapes 9-15 en utilisant les équations (III.5) et (III.6)
22. **Fin Pour**
23. Calculer la probabilité $\mathbf{p_i}$ pour chaque solution $\mathbf{x_i}$ par l'équation (II.4)
24. **La phase des abeilles Onlooker:**
25. Choisir une solution $\mathbf{x_i}$ en fonction de sa probabilité $\mathbf{p_i}$
26. Produire une nouvelle solution $\mathbf{v_i}$ par l'équation (III.1) pour un changement multi-dimensionnel ou par l'équation (III.2) pour un changement unidimensionnel
27. Evaluer $\mathbf{f(v_i)}$ et mettre $\boldsymbol{FEs = FEs + 1}$
28. Appliquer le processus de sélection gourmande
29. **La phase des Scouts:**
30. Vérifier s'il existe une solution abandonnée:
31. Alors, remplacer cette solution par une autre générée aléatoirement et mettre $\boldsymbol{FEs = FEs + 1}$
32. Mémoriser la meilleure solution trouvée jusque-là et mettre le compteur $\boldsymbol{Cycle = Cycle + 1}$
33. **Jusqu'à** atteindre le critère d'arrêt ($\boldsymbol{FEs}$ >= la valeur max de $\boldsymbol{FEs}$)

L'idée repose sur l'extension du concept d'apprentissage coopératif à la phase Onlooker avec l'introduction d'un mécanisme de recherche à règles multiples composé d'équations, de recherche locale et globale. Cette démarche conforte la description comportementale, énoncée ci-dessus, qui considère que les sous-groupes séparés d'abeilles auront à évoluer simultanément pendant les phases Employed et Onlooker, tout en assurant un échange mutuel et simultané des informations sur les sources de nourriture entre les sous-groupes d'une même catégorie.

III.2.2.1 Structure et mécanisme de recherche

Pour construire le modèle mCLABC, un mécanisme de recherche multiple associé séparément à chaque sous-groupe d'abeilles engagé dans le fourragement est d'abord défini. Les équations suivantes résument l'ensemble des règles établies à cet effet :

Règles de recherche globale :

$$v_i^j = x_i^j + \emptyset_i^j\left(x_i^j - x_k^j\right),\ i = 1,2,\dots,SN,\ i \neq k,\ j \in \{1,2,\dots,D\} \quad \text{(III.7)}$$

$$v_i^j = x_k^j + \emptyset_i^j\left(x_k^j - x_{k1}^j\right),\ i = 1,2,\dots,SN,\ i \neq k \neq k_1,\ j \in \{1,2,\dots,D\} \quad \text{(III.8)}$$

$$v_i^j = x_k^j + \emptyset_i^j\left(x_{k1}^j - x_{k2}^j\right),\ i = 1,2,\dots,SN,\ i \neq k \neq k_1 \neq k_2,\ j \in \{1,2,\dots,D\} \quad \text{(III.9)}$$

Règles de recherche locale :

$$v_i^j = x_{best}^j + \varphi_i^j\left(x_i^j - x_{mean}^j\right) + \emptyset_i^j\left(x_k^j - x_{k1}^j\right),\ i = 1,2,\dots,SN, i \neq k \neq k_1, j \in \{1,2,\dots,D\} \quad \text{(III.10)}$$

$$v_i^j = x_{best}^j + \emptyset_i^j\left(x_k^j - x_{k1}^j\right),\ i = 1,2,\dots,SN,\ i \neq k \neq k_1,\ j \in \{1,2,\dots,D\} \quad \text{(III.11)}$$

$$v_i^j = x_i^j + \varphi_i^j\left(x_i^j - x_{mean}^j\right) + \emptyset_i^j\left(x_k^j - x_{k1}^j\right),\ i = 1,2,\dots,SN,\ i \neq k \neq k_1,\ j \in \{1,2,\dots,D\} \quad \text{(III.12)}$$

où $\emptyset_i^j \in [-1,1]$, $\varphi_i^j \in [0,1]$ sont des nombres aléatoires générés pour chaque dimension à chaque instant de traitement, x_k^j, x_{k1}^j et x_{k2}^j représentent les $j^{ièmes}$ dimensions des solutions choisies aléatoirement parmi les sous-populations, les indices k, k_1, k_2, sont générés aléatoirement dans l'intervalle $[1, SN]$ et doivent être mutuellement différents de l'indice courant i, x_{best}^j désigne la $j^{ième}$ dimension de la meilleure solution globale obtenue jusque-là par les sous-populations, et x_{mean}^j est la solution moyenne correspondant à la $j^{ième}$ dimension de toutes les solutions dans les sous-populations (sous-groupes).

III.2.2.2 Procédure de recherche

A titre illustratif, la figure III.1 schématise des exemples de déplacements en deux dimensions fournis par les équations de recherche définies ci-dessus. Les équations (III.7)-(III.9) conviennent pour la recherche globale et les équations (III.10)- (III.12) pour la recherche locale. Pour incorporer le mécanisme de recherche multiple décrit par les équations (III.7)-(III.12) dans le modèle ABC de base, les abeilles Employed et les abeilles Onlooker sont d'abord catégorisées en sous-groupes. Les solutions potentielles des sous-populations sont alors générées de manière itérative dans les phases Employed et Onlooker. Conformément à la description comportementale établie, les sous-groupes d'abeilles Employed et d'abeilles Onlooker doivent évoluer différemment pendant le processus de recherche de nourriture en considérant pour chaque sous-groupe des stratégies de recherche spécifiques qui ne sont pas similaires à celles définies pour les autres sous-groupes. Selon la conception adoptée, les équations de recherche (III.7) et (III.8) seront appelées par le premier sous-groupe d'abeilles Employed et le premier sous-groupe d'abeilles Onlooker pour générer de nouvelles solutions candidates. Pour le premier sous-groupe, chaque mise à jour de solution obéirait à la règle (III.7) ou (III.8). Le choix est guidé par un mécanisme probabiliste défini en fonction de l'itération courante ($iter$) et du nombre maximal de cycles (MCN) et qui décrit par la probabilité : $\gamma = (iter/MCN)^{1/2}$. Il s'agit d'une règle non linéaire croissante, déterminée en simulation à partir de l'évaluation des capacités d'exploration et d'exploitation de la stratégie d'optimisation proposée.

Sur la base de cette définition, nous proposons de définir la règle de mise à jour sélective pour un sous-groupe d'abeille de la manière suivante :

$$\begin{aligned}
&\boldsymbol{Si}\ (random \leq \gamma)\ \boldsymbol{alors}\\
&v_i^j = x_i^j + \emptyset_i^j\left(x_i^j - x_k^j\right)\\
&\boldsymbol{Sinon}\\
&v_i^j = x_k^j + \emptyset_i^j\left(x_k^j - x_{k1}^j\right)\\
&\boldsymbol{FinSi}
\end{aligned} \qquad \text{(III.13)}$$

D'après l'équation (III.13), il est clair que la probabilité d'utiliser l'une des deux règles de recherche s'établit en fonction du nombre d'itérations. La forme non linéaire de la probabilité est déterminée à travers plusieurs simulations en considérant le taux de convergence et la qualité des solutions fournies à l'issue du processus de recherche. De manière équivalente et sur la base de la même définition de probabilité (III.13), les équations de recherche aléatoire (III.9) et (III.10) guidées par la meilleure solution globale sont utilisées pour le deuxième

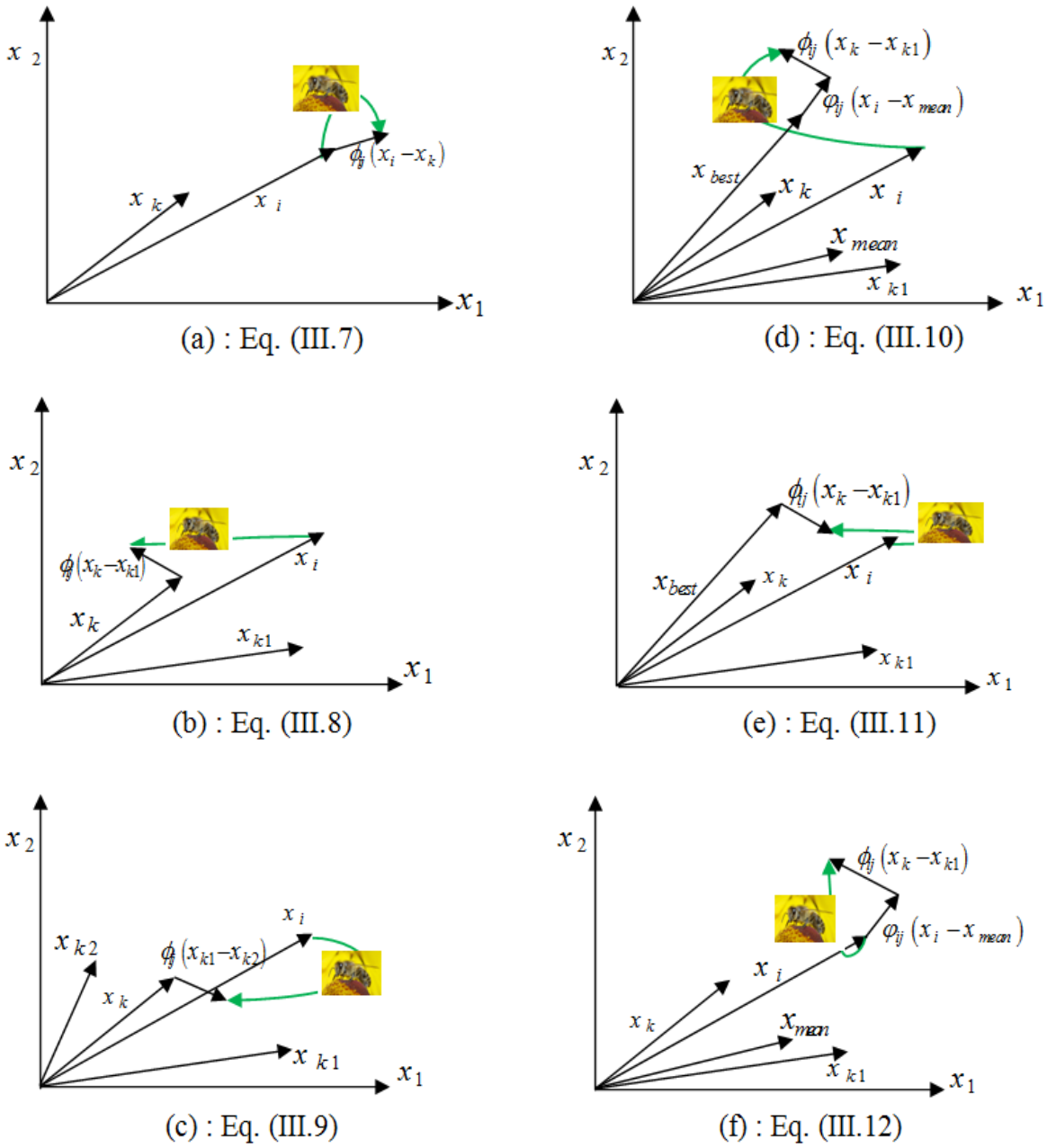


Figure III.1 : Les différentes possibilités de mouvement d'une abeille selon les équations de recherche définies dans le modèle mCLABC dans un espace paramétrique de 2 dimensions.

sous-groupe d'abeilles Employed et le deuxième sous-groupe d'abeilles Onlooker. Alors que le troisième sous-groupe utilise les équations (III.11) et (III.12) pour faire évoluer les solutions candidates. L'équation (III.11) effectue la mise à jour des solutions dans une direction aléatoire guidée par la meilleure solution courante trouvée, alors que l'équation de recherche (III.12) est orientée vers la moyenne.

D'autre part, une stratégie pour sélectionner le nombre de paramètres à faire évoluer à chaque mise à jour de solutions est incorporée au modèle mCLABC dans les deux phases Employed et Onlooker. Pour ce faire, les abeilles Employed sont gérées à commencer la recherche en changeant une seule dimension aléatoire. Ensuite, lorsque le nombre d'itérations atteint la valeur $iter = \frac{1}{5}MCN$, l'algorithme devrait optimiser de manière aléatoire un ou tous les paramètres simultanément. Vers la fin du processus itératif, quand le nombre d'itérations est entre $iter = \frac{4}{5}MCN$ et $iter = MCN$, les abeilles Employed font évoluer leurs solutions en changeant tous les paramètres de la solution. Par la suite, après l'achèvement de la phase Employed et compte tenu du processus de *sélection gourmande*, les abeilles Onlooker commenceraient leur recherche de solutions à partir d'un déplacement aléatoire unidimensionnel. Puis, lorsque $iter = \frac{1}{5}MCN$, les abeilles Onlooker procèderaient aléatoirement à la mise à jour de toutes les dimensions de la solution ou bien une seule dimension choisie aléatoirement jusqu'à la fin du processus de recherche ($iter = MCN$). La stratégie sélective d'altération des sources de nourriture du modèle mCLABC est illustrée en Figure III.2.

III.2.2.3 Equilibre entre exploitation et exploration

La stratégie de recherche multiple proposée est conçue avec une possibilité d'appliquer toutes les règles de mise à jour à n'importe quel niveau du processus de recherche. Le mécanisme établi à cet effet permet d'assurer une recherche globale au début du processus afin d'améliorer l'exploration et une recherche locale à travers les générations pour accroître l'exploitation. En ce qui concerne la définition de probabilité, les équations de recherche (III.8), (III.9) et (III.12) auront plus de chance d'être appelées au début du processus d'optimisation, ce qui pourrait assurer la diversité. Ensuite, toutes les équations de recherche globale (III.7), (III.8) et (III.9) et toutes les équations de recherche locale (III.10), (III.11) et (III.12) seront utilisées de manière aléatoire pour intensifier la recherche. En conséquence, l'exploration globale et l'exploitation locale sont équilibrées à ce niveau. Par la suite, et selon les valeurs de la probabilité γ, toutes les stratégies de recherche pourraient être utilisées, mais

cette fois-ci, les équations (III.7), (III.10) et (III.11) auront plus de chance d'être impliquées ce qui pourrait augmenter la capacité d'exploitation, améliorer la convergence et préserver la capacité d'exploration. Le taux de sélection associé à chaque règle de mise à jour des solutions pendant le processus de recherche est illustré en figure III.3.

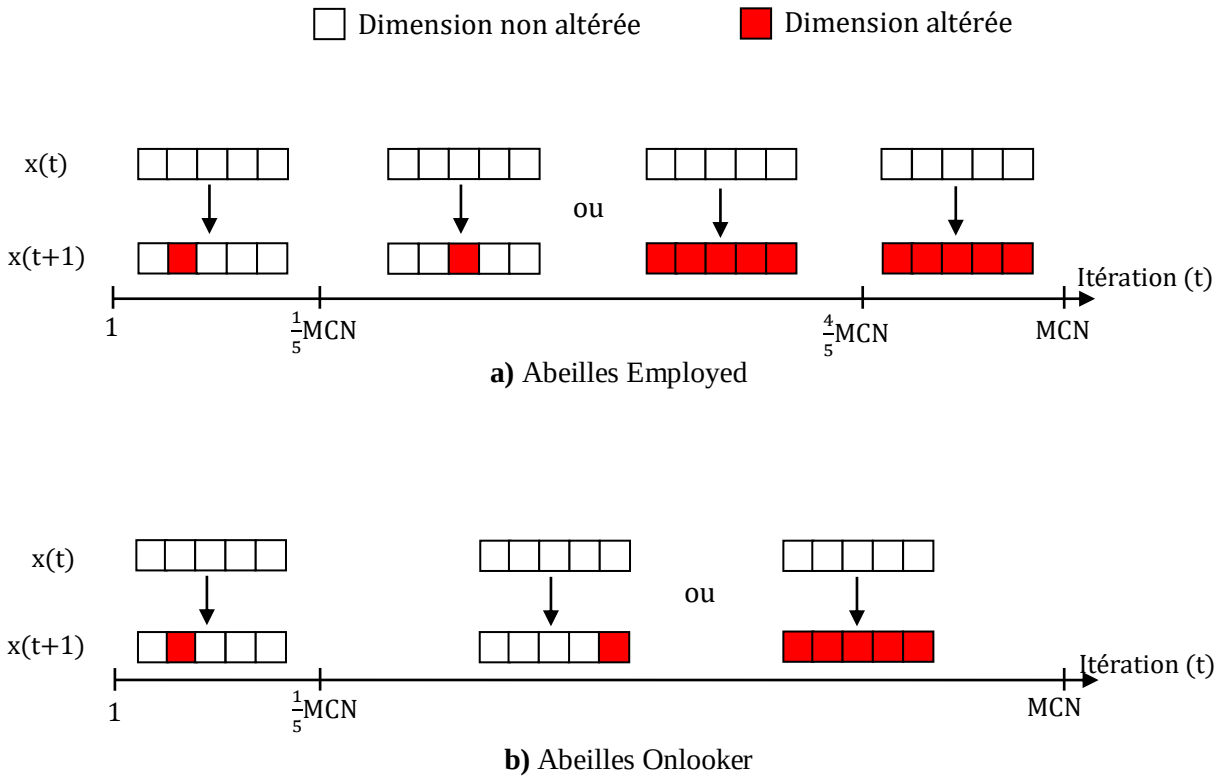


Figure III.2 : Stratégie sélective d'altération des sources de nourriture du modèle mCLABC.

A travers cette description, nous désirions apporter une caractérisation stochastique projetée sur le comportement naturel de recherche de nourriture chez les essaims d'abeilles, qui pourrait en toute évidence paraître très sélectif en ce qui concerne l'altération des sources de nourriture. Cette caractérisation est explicitée par le modèle mCLABC décrit par l'algorithme III.2. Les pseudo-codes décrivant les processus de recherche de nourriture associés aux sous-groupes Employed et Onlooker sont donnés par les algorithmes III.3 et III.4, respectivement.

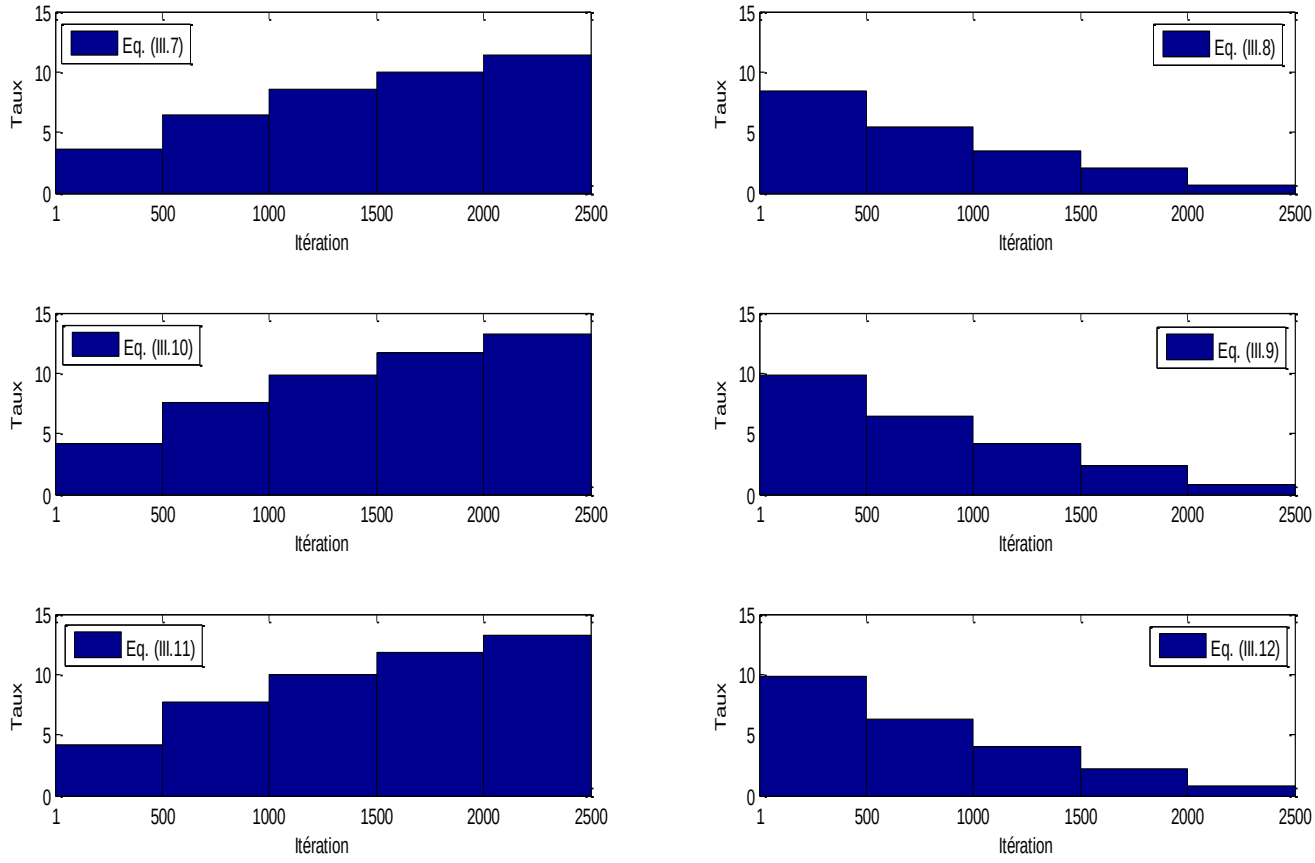


Figure III.3 : Taux de sélection des équations de recherche dans le modèle mCLABC.

III.3 Analyse de performance de la stratégie d'optimisation proposée

La validation d'une méthode d'optimisation repose nécessairement sur une étude de performance approfondie s'appuyant sur une analyse comparative de résultats obtenus moyennant d'autres méthodes, ainsi que sur des tests statistiques appropriés. De ce fait, l'analyse de performance que nous avions menée pour les deux variantes CLABC et mCLABC proposées comprend les cas d'étude suivants :

- Optimisation de fonctions numériques standards.
- Optimisation de fonctions complexes (composites) catégorisées sous les labels CEC'05 et CEC'13.

Pour ces deux problèmes, une étude en simulation étendue est dressée illustrant des résultats comparant les performances de la stratégie proposée au modèle ABC de base et à ses principales variantes, ainsi qu'à d'autres métaheuristiques avancées. L'étude comparative est suivie d'une analyse statistique de performance commune à ce genre de problèmes.

Algorithme III.2 : Pseudo-code de l'algorithme mCLABC

1. Initialisation
2. Générer une population initiale de $\boldsymbol{SN}$ solutions par l'équation (II.1)
3. Evaluer la population initiale par l'équation (II.2)
4. $\boldsymbol{Cycle = 1}$
5. **Répéter**
6. **La phase des abeilles Employed:**
7. Envoyer le sous-groupe 1 des abeilles Employed et exécuter le processus décrit par l'algorithme III.3 en utilisant les équations (III.7) et (III.8)
8. Envoyer le sous- groupe 2 des abeilles Employed et exécuter le processus décrit par l'algorithme III.3 en utilisant les équations (III.9) et (III.10)
9. Envoyer le sous- groupe 3 des abeilles Employed et exécuter le processus décrit par l'algorithme III.3 en utilisant les équations (III.11) et (III.12)
10. Appliquer le processus de la sélection gourmande
11. Calculer la probabilité $\boldsymbol{p_i}$ de chaque solution $\boldsymbol{x_i}$ par l'équation (II.4)
12. **La phase des abeilles Onlooker:**
13. Envoyer le sous- groupe 1 des abeilles Onlooker et exécuter le processus décrit par l'algorithme III.4 en utilisant les équations (III.7) et (III.8)
14. Envoyer le sous- groupe 2 des abeilles Onlooker et exécuter le processus décrit par l'algorithme III.4 en utilisant les équations (III.9) et (III.10)
15. Envoyer le sous- groupe 3 des abeilles Onlooker et exécuter le processus décrit par l'algorithme III.4 en utilisant les équations (III.11) et (III.12)
16. Appliquer le processus de la sélection gourmande
17. **La phase des Scouts:**
18. S'il existe une solution abandonnée, remplacer cette solution par une autre générée aléatoirement
19. Mémoriser la meilleure solution trouvée
20. $\boldsymbol{Cycle = Cycle + 1}$
21. **Jusqu'à** atteindre le critère d'arrêt

III.3.1 Problèmes d'optimisation de fonctions numériques standards

Pour cette première phase d'analyse de performance, nous considérons des problèmes d'optimisation numériques définis sur 22 fonctions benchmark de différents niveaux de complexité et de multimodalité. Les fonctions considérées sont décrites en Annexe A. Les tests effectués sont menés sur les fonctions suivantes :

- Les fonctions de faible dimension : F1-F20 avec $D = 15$, F21 et F22 avec $D = 50$.
- Les fonctions de moyenne dimension : F1-F20 avec D = 30, F21 et F22 avec $D = 100$.
- Les fonctions de grande dimension : F1-F20 avec D = 60, F21 et F22 avec $D = 200$.

Algorithme III.3 : Pseudo-code du processus de recherche de nourriture (fourragement) associé aux sous-groupes d'abeilles Employed dans le modèle mCLABC

1. **Sous-groupe d'abeilles Employed**
2. mettre $\boldsymbol{\gamma} = (\boldsymbol{Cycle/MaxCycle})\text{^}(\boldsymbol{1/2})$
3. **Pour** toutes les sources de nourriture $\boldsymbol{SN_{employed}}$ visitées par les abeilles Employed
4. Appliquer la stratégie évolutive de sélection des paramètres
5. **Si** (tous les paramètres doivent être modifiés)
6. **Si** ($\boldsymbol{rand(0,1) <= \gamma}$)
7. Produire une nouvelle source de nourriture $\boldsymbol{V_i}$ pour l'abeille Employed en utilisant la règle de recherche spécifiée (SRule1)*
8. **Si non**
9. Produire une nouvelle source de nourriture $\boldsymbol{V_i}$ pour l'abeille Employed en utilisant la règle de recherche spécifiée (SRule2)*
10. **Fin Si**
11. **Si non**
12. **Si** ($\boldsymbol{rand(0,1) <= \gamma}$)
13. Produire une nouvelle source de nourriture $\boldsymbol{V_i}$ pour l'abeille Employed en modifiant un paramètre aléatoire à l'aide de la règle de recherche spécifiée (SRule1)*
14. **Si non**
15. Produire une nouvelle source de nourriture $\boldsymbol{V_i}$ pour l'abeille Employed en modifiant un paramètre aléatoire à l'aide de la règle de recherche spécifiée (SRule2)*
16. **Fin Si**
17. **Fin Si**
18. **Fin Pour**

(*) SRule1: désigne l'équation (III.7), (III.10) ou (III.11) associée au premier, deuxième et troisième sous-groupe d'abeilles, respectivement.
(*) SRule2: désigne l'équation (III.8), (III.9) ou (III.12) associée au premier, deuxième et troisième sous-groupe d'abeilles, respectivement.

Pour l'ensemble des cas étudiés, le critère d'arrêt est défini en fonction du nombre maximal d'évaluations de la fonction objectif (Max_FEs). Le paramètre Max_FEs est fixé à 50.000 pour les tests en faible dimension, 100.000 pour la moyenne dimension, et 200.000 pour les tests de grande dimension. La taille de la colonie (CS) est fixée à $CS = 40$, la valeur $limit$ est donnée par $limit = D * SN$, où D représente la dimension du problème et SN est le nombre de sources de nourriture (solutions), avec $SN = CS/2$.

III.3.1.1 Comparaison au modèle ABC de base

Les algorithmes CLABC, mCLABC et ABC sont testés indépendamment sur chacune des fonctions avec un nombre total d'essais égalant 30. Les tableaux III.1, III.2 et III.3 présentent

les résultats obtenus exprimés en termes d'erreur moyenne (Moy.) et d'écart-type (Éc.-Type) évalués sur les valeurs du minimum global calculés sur 30 essais.

Algorithme III.4 : Pseudo-code du processus de recherche de nourriture (fourragement) associé aux sous-groupes d'abeilles Onlooker dans le modèle mCLABC

1. **Sous-groupe d'abeilles Onlooker**
2. $\boldsymbol{t = 0, i = 1}$
3. **Tant que** ($\boldsymbol{t <= SN_{employed}}$) **Faire**
4. **Si** ($\boldsymbol{rand(0,1) < pi}$) Alors
5. Appliquer la stratégie évolutive de sélection des paramètres
6. **Si** (tous les paramètres doivent être modifiés)
7. **Si** ($\boldsymbol{rand(0,1) <= \gamma}$)
8. Produire une nouvelle source de nourriture $\boldsymbol{V_i}$ pour l'abeille Onlooker en utilisant la règle de recherche spécifiée (SRule1)*
9. **Sinon**
10. Produire une nouvelle source de nourriture $\boldsymbol{V_i}$ pour l'abeille Onlooker en utilisant la règle de recherche spécifiée (SRule2)*
11. **Fin Si**
12. **Sinon**
13. **Si** ($\boldsymbol{rand(0,1) <= \gamma}$)
14. Produire une nouvelle source de nourriture $\boldsymbol{V_i}$ pour l'abeille Onlooker en modifiant un paramètre aléatoire à l'aide de la règle de recherche spécifiée (SRule1)*
15. **Sinon**
16. Produire une nouvelle source de nourriture $\boldsymbol{V_i}$ pour l'abeille Onlooker en modifiant un paramètre aléatoire à l'aide de la règle de recherche spécifiée (SRule2)*
17. **Fin Si**
18. **Fin Si**
19. Appliquer le processus de sélection gourmande
20. $\boldsymbol{t = t + 1}$
21. **Fin Si**
22. $\boldsymbol{i = i + 1}$, **Si** ($\boldsymbol{i = (SN_{onlooker}) + 1}$), **Alors,** $\boldsymbol{i = 1}$
23. **Fin**

(*) SRule1: désigne l'équation (III.7), (III.10) ou (III.11) associée au premier, deuxième et troisième sous-groupe d'abeilles, respectivement.
(*) SRule2: désigne l'équation (III.8), (III.9) ou (III.12) associée au premier, deuxième et troisième sous-groupe d'abeilles, respectivement.

Les courbes de convergence des algorithmes comparés, choisies pour des problèmes de différentes dimensionnalités et différentes caractéristiques parmi les cas considérés sont illustrées en figure III.4. Sur ce plan de convergence, les résultats obtenus montrent clairement la supériorité des variantes CLABC et mCLABC proposées par rapport au modèle ABC de base. En outre, il a y lieu de remarquer que la variante mCLABC convergence très

rapidement par rapport aux modèles ABC et CLABC, en particulier pour les fonctions de moyenne et de grande dimension.

Les résultats résumés dans les tableaux III.1-III.3 renseignent également de la performance de la stratégie proposée au même titre que l'étude de convergence présentée par la figure III.4.

Tableau III.1 : Comparaison des performances des algorithmes mCLABC, CLABC et ABC sur des problèmes d'optimisation des fonctions F1-F20 (D = 15) et des fonctions F21-F22 (D = 50).

Fonction	ABC		CLABC		mCLABC	
	Moy.	Éc.-Type	Moy.	Éc.-Type	Moy.	Éc.-Type
F1	2.36e-16	6.35e-17	**1.03e-66**	**4.91e-66**	1.72e-62	3.04e-62
F2	2.27e-16	3.66e-17	**2.73e-63**	**1.24e-62**	1.35e-58	2.37e-58
F3	2.40e-16	4.61e-17	**8.89e-70**	**2.85e-69**	5.59e-64	9.97e-64
F4	2.14e-17	8.42e-18	**8.31e-135**	**3.17e-134**	7.47e-122	1.89e-121
F5	5.48e-16	7.89e-17	**1.50e-38**	**6.95e-38**	5.71e-34	6.82e-34
F6	2.21e-01	7.15e-02	1.56e-03	3.14e-03	**1.05e-04**	**3.95e-04**
F7	**0**	**0**	**0**	**0**	**0**	**0**
F8	6.36e-17	1.63e-17	7.52e-122	4.06e-121	**2.24e-122**	**5.44e-122**
F9	2.59e-02	9.84e-03	**3.15e-03**	**1.11e-03**	3.69e-03	1.52e-03
F10	6.07e-02	7.81e-02	**1.30e-03**	**5.07e-03**	1.75e-01	9.34e-01
F11	**0**	**0**	**0**	**0**	**0**	**0**
F12	**0**	**0**	**0**	**0**	**0**	**0**
F13	2.63e-03	4.57e-03	**2.69e-04**	**1.46e-03**	1.48e-03	4.50e-03
F14	5.01e-03	4.32e-03	3.89e-03	5.32e-03	**2.91e-03**	**4.53e-03**
F15	1.84e-14	3.36e-15	7.14e-15	1.97e-15	**7.05e-15**	**1.85e-15**
F16	2.42e-16	4.70e-17	**3.14e-32**	**1.11e-47**	**3.14e-32**	**1.11e-47**
F17	2.33e-16	4.71e-17	**1.34e-32**	**5.57e-48**	**1.34e-32**	**5.57e-48**
F18	3.85e-09	5.35e-09	1.60e-12	6.79e-12	**9.34e-31**	**2.24e-30**
F19	1.73e-16	6.00e-17	1.34e-31	1.77e-46	**1.34e-31**	**6.68e-47**
F20	**0**	**0**	1.17e-07	6.42e-07	**0**	**0**
F21	-78.33	4.90e-04	-78.33	7.13e-14	**-78.33**	**5.21e-14**
F22	-47.22	3.41e-01	-48.30	2.69e+01	**-49.11**	**1.15e-01**

Tableau III.2 : Comparaison des performances des algorithmes mCLABC, CLABC et ABC sur des problèmes d'optimisation des fonctions F1-F20 (D = 30) et des fonctions F21-F22 (D = 100).

Fonction	ABC		CLABC		mCLABC	
	Moy.	Éc.-Type	Moy.	Éc.-Type	Moy.	Éc.-Type
F1	5.90e-16	9.58e-17	3.71e-57	1.89e-56	**8.08e-58**	**1.14e-57**
F2	4.98e-16	8.79e-17	2.73e-53	8.74e-53	**1.15e-53**	**3.94e-53**
F3	5.53e-16	9.18e-17	7.33e-58	2.86e-57	**1.47e-58**	**1.97e-58**
F4	3.39e-17	1.15e-17	**3.74e-128**	**2.60e-127**	2.25e-117	1.23e-116
F5	1.31e-15	1.47e-16	**3.71e-34**	**1.55e-33**	7.66e-32	9.83e-32
F6	3.86	1.56	3.59	1.21	**8.13e-01**	**5.75e-01**
F7	**0**	**0**	**0**	**0**	**0**	**0**
F8	2.17e-16	5.35e-17	2.06e-99	1.05e-98	**1.19e-110**	**7.39e-110**
F9	6.02e-02	1.49e-02	8.49e-03	2.90e-03	**7.56e-03**	**3.27e-03**
F10	6.25 e-02	9.61e-02	8.12e-03	3.54e-02	**5.88e-03**	**1.66e-02**
F11	**0**	**0**	**0**	**0**	**0**	**0**
F12	2.35e-13	1.28e-12	**0**	**0**	**0**	**0**
F13	9.57e-13	5.24e-12	**0**	**0**	**0**	**0**
F14	5.73e-03	4.18e-03	**1.30e-03**	**3.36e-03**	2.72e-03	4.41e-03
F15	4.40e-14	5.42e-15	**1.42e-14**	**3.56e-15**	1.59e-14	3.19e-15
F16	5.22e-16	7.14e-17	1.65e-32	3.19e-33	**1.57e-32**	**5.57e-48**
F17	5.16e-16	1.03e-16	**1.34e-32**	**5.57e-48**	**1.34e-32**	**5.57e-48**
F18	2.45e-07	3.99e-07	2.27e-10	6.45e-10	**1.90e-29**	**4.85e-29**
F19	4.43e-16	8.47e-17	2.55e-31	4.55e-31	**1.34e-31**	**6.68e-47**
F20	**0**	**0**	4.34e-06	1.56e-05	**0**	**0**
F21	-78.3308	3.70e-03	-78.3323	1.15e-10	**-78.3323**	**2.64e-13**
F22	-93.72	9.01e-01	-95.73	7.46e-01	**-97.22**	**3.11e-01**

Tableau III. 3 : Comparaison des performances des algorithmes mCLABC, CLABC et ABC sur des problèmes d'optimisation des fonctions F1-F20 (D = 60) et des fonctions F21-F22 (D = 200).

Fonction	ABC		CLABC		mCLABC	
	Moy.	Éc.-Type	Moy.	Éc.-Type	Moy.	Éc.-Type
F1	1.31e-15	1.27e-16	1.14e-46	6.75e-46	**2.68e-53**	**3.21e-53**
F2	1.11e-15	1.69e-16	6.60e-43	1.63e-42	**8.50e-50**	**1.25e-49**
F3	1.29e-15	1.49e-16	1.04e-47	2.04e-47	**4.50e-54**	**4.47e-54**
F4	4.77e-17	1.62e-17	1.09e-105	7.68e-105	**4.57e-108**	**2.25e-107**
F5	2.98e-15	1.96e-16	5.97e-29	1.51e-28	**1.00e-29**	**1.56e-29**
F6	2.06e+01	4.18e+00	16.50	2.90	**7.19e+00**	**1.55e+00**
F7	**0**	**0**	**0**	**0**	**0**	**0**
F8	5.29e-16	8.03e-17	3.16e-81	7.17e-81	**9.11e-102**	**3.49e-101**
F9	1.28e-01	2.26e-02	2.42e-02	6.58e-03	**1.48 e-02**	**5.13 e-03**
F10	1.03e-01	2.34e-01	2.28e-02	8.56e-02	**1.87 e-02**	**5.72 e-02**
F11	**0**	**0**	**0**	**0**	**0**	**0**
F12	**0**	**0**	**0**	**0**	**0**	**0**
F13	2.47e-04	1.35e-03	**0**	**0**	**0**	**0**
F14	4.25e-03	4.09e-03	2.91e-03	4.69e-03	**2.59e-03**	**4.37e-03**
F15	1.04e-13	8.97e-15	**3.54e-14**	**5.83e-15**	4.10e-14	6.34e-15
F16	1.27e-15	1.16e-16	8.64e-33	2.13e-33	**7.85e-33**	**2.78e-48**
F17	1.30e-15	1.51e-16	**1.34e-32**	**5.57e-48**	**1.34e-32**	**5.57e-48**
F18	2.66e-05	9.86e-05	9.58e-10	1.12e-09	**4.87e-28**	**7.57e-28**
F19	1.10e-15	1.44e-16	1.48e-31	2.96e-32	**1.34e-31**	**6.68e-47**
F20	1.89e-15	4.91e-15	8.67e-05	3.38e-04	**0**	**0**
F21	-78.31	4.33e-02	-78.32	1.20e-02	**-78.33**	**2.32e-12**
F22	-186.712	1.48	190	1.48	**-192.635**	**4.35e-01**

En effet, comme indiqué dans le tableau III.1, les modèles mCLABC et CLABC obtiennent les meilleurs résultats en comparaison à l'algorithme ABC de base. Cependant, la variante CLABC surpasse la variante mCLABC sur les fonctions unimodales de faible dimension F1, F2, F3, F5, F9 et F10, tandis que l'algorithme mCLABC présente des résultats qui sont bien meilleurs que ceux obtenus par l'algorithme CLABC sur la plupart des fonctions multimodales de faible dimension F14, F15, F18, F19, F20 et F22 qui sont des fonctions numériques plutôt complexes.

D'autre part, et d'après le tableau III.2, nous pouvons remarquer que l'approche mCLABC arrive à réaliser des résultats significativement meilleurs que les autres sur la majorité des fonctions unimodales et multimodales de moyenne dimension. La variante CLABC montre des performances meilleures sur les seules fonctions F4, F5 et F14, avec une amélioration légère pour la fonction F15.

En outre, les résultats indiqués dans le tableau III.3 affichent une nette supériorité de l'approche mCLABC sur des problèmes de grande dimension regroupant des fonctions séparables, non séparables, unimodales et multimodales. Il convient de noter que les algorithmes ABC et CLABC ne parviennent pas à surpasser l'algorithme mCLABC dans ces cas d'étude à l'exception de la fonction F15, où l'on note que l'algorithme CLABC réalise un résultat légèrement supérieur en terme d'erreur moyenne de la fonction.

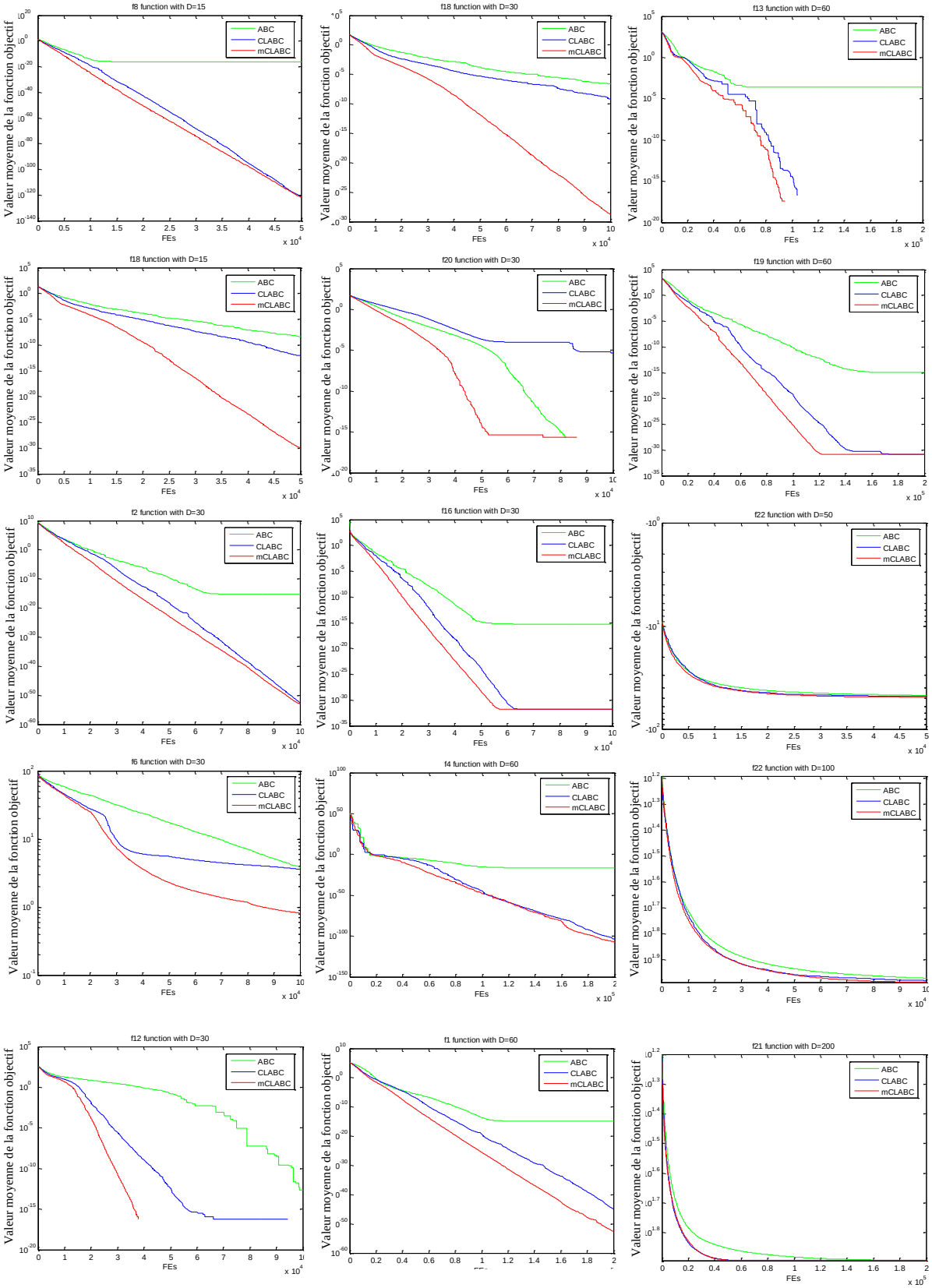


Figure III.4 : Les courbes de convergence des algorithmes ABC, CLABC et mCLABC illustrant 15 fonctions numériques de différentes dimensionnalités.

Cette étude comparative de performance menée par rapport à la version originale du modèle ABC appuie clairement l'efficacité du processus de recherche à règles multiples fondé sur la stratégie d'apprentissage coopératif que nous proposons d'intégrer au modèle de base à travers les algorithmes CLABC et mCLABC.

III.3.1.2 Comparaison aux variantes du modèle ABC de base

Dans cette deuxième phase d'analyse de performance de la stratégie proposée, une étude comparative est établie entre le modèle mCLABC et cinq autres variantes de l'algorithme ABC développées dans la littérature. Les variantes en question sont: l'algorithme CABC (ABC based on modified search strategy) [84], l'algorithme ABCbest (Global best ABC) [78], l'algorithme OGABC (Global-best-guided ABC based on orthogonal learning strategy) [84], l'algorithme MuABC (ABC with multiple search strategies) [80], et l'algorithme ABCVSS (ABC with variable search strategy) [81]. Les résultats comparés sont repris directement depuis les références [81] et [80]. Les tableaux III.4, III.5 et III.6 résument l'ensemble des cas étudiés pour les fonctions de faible dimension, de moyenne dimension et de grande dimension. Les résultats sont indiqués en termes d'erreur moyenne (Moy.) et d'écart-type (Éc.-Type) évalués sur 30 essais.

En analysant ces résultats, il en ressort d'après le tableau III.4 que l'approche mCLABC réalise les mêmes performances que les variantes comparées sur les fonctions F7, F11, F12, F16, F20 et F21. Cependant, la supériorité du modèle mCLABC est nettement visible pour les fonctions F1, F2 F3 F4 F5 et F6. Pour le reste des fonctions testées, la stratégie que nous proposons affiche une efficacité computationnelle établie sur au moins trois variantes comparées du modèle ABC.

Pour les fonctions de dimension moyenne, les résultats obtenus sont présentés dans le tableau III.5, où il apparait clairement que l'algorithme mCLABC confirme sa performance pour les problèmes d'optimisation des fonctions F1, F2, F4, F5, F10 et F15. Une équivalence est cependant déduite pour les fonctions F7, F11 et F21. Pour le reste des fonctions, le modèle mCLABC montre une efficacité égale ou supérieure à au moins trois autres variantes.

Pour la plupart des fonctions de grande dimension montrées sur le tableau III.6, les résultats du modèle mCLABC exprimés en termes de valeurs moyennes de la fonction objectif sont nettement meilleurs à l'exception de cinq problèmes étudiés où l'on constate une légère supériorité des variantes comparés. Néanmoins, il est important de noter que la solution optimale obtenue pour la fonction multimodale F20 n'a pu être trouvée que par le modèle mCLABC proposé.

Pour conclure cette deuxième phase d'analyse de performance, nous constatons que la stratégie mCLABC proposée se place en tête de classement pour les problèmes d'optimisation définis sur les fonctions unimodales de faible, moyenne et grande dimension. Pour les fonctions multimodales, les variantes MuABC et ABCVSS affichent une performance meilleure pour les problèmes de faible dimension, tandis que le modèle mCLABC est autant plus performant pour les problèmes définis sur des fonctions multimodales de moyenne et de grande dimension.

III.3.1.3 Analyse statistique de performance de la stratégie proposée

L'analyse de performance d'une méthode d'optimisation basée sur l'évaluation des erreurs moyennes et de leurs déviations est assez représentative et concluante quant à la validité de l'approche. Néanmoins, pour certains résultats comparatifs équivalents, il est parfois nécessaire d'appuyer l'analyse de performance par des tests statistiques supplémentaires en vue d'une meilleure visibilité sur la convergence de l'approche.

Les résultats comparatifs des fonctions F7 et F11 des tableaux III.4, III.5 et III.6 en représentent un exemple typique pour la stratégie proposée. Pour une meilleure interprétation, une analyse statistique est appliquée aux résultats obtenus par l'approche proposée mCLABC et ceux obtenus par les variantes du modèle ABC décrits dans le paragraphe III.3.1.2. Elle se fond sur deux critères d'évaluation statistiques qui sont :

Le critère AVEN (Average number of fitness evaluations) qui désigne la moyenne du nombre d'évaluations de la fonction objectif, et le critère SR (Success rate) qui représente le taux de réussite de l'approche. Les résultats de l'analyse statistique de la stratégie mCLABC proposée sont résumés dans les tableaux III.7–III.9.

Les résultats des algorithmes comparés sont tirés directement de la référence [80]. Une première constatation affiche une nette supériorité du modèle mCLABC par rapport aux algorithmes CABC et ABCbest sur l'ensemble des fonctions tests considérées. L'algorithme mCLABC achève un taux de réussite de 100% pour toutes les fonctions, sauf pour F13 ($D = 15$), il est cependant compétitif à l'algorithme MuABC, introduit récemment, et qui présente un taux de réussite de 100% pour toutes les fonctions, à l'exception de F9 ($D = 60$). Cependant, l'algorithme MuABC nécessitera un nombre élevé d'évaluations pour afficher un taux de réussite de 100%. Les résultats de cette analyse statistique confortent les conclusions de l'analyse de performance de la stratégie proposée décrites ci-dessus, ce qui avance une parfaite visibilité quant à la validité et l'efficacité du modèle d'optimisation proposé.

Tableau III.4 : Comparaison des performances des algorithmes mCLABC, ABCVSS, MuABC, OGABC, ABCbest et CABC sur des problèmes d'optimisation des fonctions F1-F20 (D = 15) et des fonctions F21-F22 (D = 50).

Fonction	CABC		ABCbest		OGABC		MuABC		ABCVSS		mCLABC	
	Moy.	Éc.-Type	Moy.	Éc.-Type	Moy.	Éc.-Type	Moy.	Éc.-Type	Moy.	Éc.-Type	Moy.	Éc.-Type
F1	4.26e-36	4.76e-36	1.27e-34	1.03e-34	3.12e-33	4.85e-33	1.34e-51	1.75e-51	3.01e-61	1.49e-60	**1.72e-62**	**3.04e-62**
F2	2.32e-32	3.03e-32	2.59e-31	2.54e-31	2.57e-29	5.56e-29	2.47e-48	2.43e-48	1.21e-53	6.64e-53	**1.35e-58**	**2.37e-58**
F3	3.51e-37	6.22e-37	3.17e-36	2.48e-36	2.02e-34	3.04e-34	9.77e-53	5.81e-53	4.91e-60	1.90e-59	**5.59e-64**	**9.97e-64**
F4	7.98e-50	2.35e-49	2.00e-56	4.43e-56	1.40e-46	4.72e-46	4.81e-73	7.81e-73	2.67e-63	1.46e-62	**7.47e-122**	**1.89e-121**
F5	3.11e-19	3.27e-19	3.92e-19	1.04e-19	1.09e-17	8.70e-18	7.36e-28	5.39e-28	6.14e-29	1.93e-28	**5.71e-34**	**6.82e-34**
F6	3.91e-01	1.06e-01	2.06e-01	5.93e-02	5.41e-02	1.54e-02	3.81e-03	1.15e-03	3.13e-03	3.68e-03	**1.05e-04**	**3.95e-04**
F7	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
F8	-	-	-	-	-	-	-	-	-	-	7.47e-122	1.89e-121
F9	1.62e-02	7.26e-03	6.70e-03	1.91e-03	**3.64e-03**	**1.33e-03**	4.82e-03	1.25e-03	9.28e-03	3.13e-03	3.69e-03	1.52e-03
F10	1.58e-01	1.37e-01	2.48e-00	4.60e-00	5.32e-01	9.43e-01	**2.12e-02**	**6.35e-02**	2.57e-01	7.91e-01	1.75e-01	9.34e-01
F11	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
F12	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
F13	9.39e-10	4.69e-09	9.34e-03	1.41e-02	9.85e-04	2.75e-03	**8.36e-11**	**1.44e-11**	9.04e-04	2.78e-03	1.48e-03	4.50e-03
F14	-	-	-	-	-	-	-	-	-	-	2.91e-03	4.53e-03
F15	1.03e-14	3.06e-15	1.06e-14	2.51e-15	3.49e-14	1.21e-14	**1.42e-15**	**2.65e-15**	8.88e-15	2.91e-15	7.05e-15	1.85e-15
F16	**3.14e-32**	**1.11e-47**	**3.14e-32**	**0**	**3.14e-32**	**1.11e-47**	**3.14e-32**	**0**	**3.14e-32**	**1.11e-47**	**3.14e-32**	**1.11e-47**
F17	**1.34e-32**	**5.58e-48**	**1.34e-32**	**0**	1.46e-32	2.42e-33	**1.34e-32**	**0**	**1.34e-32**	**5.57e-48**	**1.34e-32**	**5.57e-48**
F18	4.57e-20	4.71e-20	3.99e-19	4.34e-19	1.26e-16	3.36e-16	3.70e-29	6.41e-29	**3.30e-32**	**1.63e-31**	9.34e-31	2.24e-30
F19	**1.34e-31**	**2.23e-47**	**1.34e-31**	**0**	3.08e-30	8.34e-30	**1.34e-31**	**0**	**1.34e-31**	**6.68e-47**	**1.34e-31**	**6.68e-47**
F20	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
F21	**-7.83e+01**	**9.59e-10**	**-78.3323**	**1.66e-08**	**-7.83e+01**	**2.81e-11**	**−78.3323**	**4.92e-14**	**-7.83e+01**	**1.51e-12**	**-78.33**	**5.21e-14**
F22	-4.87e+01	2.06e-01	-48.0990	3.59e-01	-4.79e+01	2.72e-01	−49.2637	7.52e-02	**-4.94e+01**	**2.32e-01**	-49.11	1.15e-01

Tableau III.5 : Comparaison des performances des algorithmes mCLABC, ABCVSS, MuABC, OGABC, ABCbest et CABC sur des problèmes d'optimisation des fonctions F1-F20 (D = 30) et des fonctions F21-F22 (D = 100).

Fonction	CABC		ABCbest		OGABC		MuABC		ABCVSS		mCLABC	
	Moy.	Éc.-Type	Moy.	Éc.-Type	Moy.	Éc.-Type	Moy.	Éc.-Type	Moy.	Éc.-Type	Moy.	Éc.-Type
F1	5.41e-35	7.02e-35	5.30e-32	3.79e-32	4.69e-38	5.27e-38	3.30e-50	1.77e-50	1.20e-57	5.01e-57	**8.08e-58**	**1.14e-57**
F2	4.67e-31	6.68e-31	4.27e-28	6.66e-28	5.38e-34	9.25e-34	3.33e-47	2.63e-47	2.44e-52	1.34e-51	**1.15e-53**	**3.94e-53**
F3	7.20e-36	7.79e-36	7.33e-33	3.18e-33	3.42e-38	1.02e-37	2.95e-51	3.87e-51	**9.53e-61**	**4.52e-60**	1.47e-58	1.97e-58
F4	3.24e-48	7.87e-48	3.24e-57	7.25e-57	7.36e-57	2.60e-56	3.30e-70	5.72e-70	1.29e-78	5.77e-78	**2.25e-117**	**1.23e-116**
F5	1.43e-18	5.68e-19	2.91e-17	1.01e-17	5.33e-20	3.13e-20	1.76e-27	8.16e-28	1.47e-27	7.90e-27	**7.66e-32**	**9.83e-32**
F6	6.01e+00	8.80e-01	4.75e+00	1.02e+00	5.84e-01	1.83e-01	2.65e+00	4.91e-02	**5.09e-01**	**3.55e-01**	8.13e-01	5.75e-01
F7	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
F8	-	-	-	-	-	-	-	-	-	-	1.19e-110	7.39e-110
F9	4.98e-02	1.78e-02	3.80e-02	2.25e-03	**5.09e-03**	**2.07e-03**	2.79e-02	2.51e-03	2.46e-02	9.49e-03	7.56e-03	3.27e-03
F10	1.96e-01	1.36e-01	2.53e+01	3.27e+01	1.10e+00	1.74e+00	1.63e-01	2.46e-01	4.92e-01	1.15e+00	**5.88e-03**	**1.66e-02**
F11	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
F12	**0**	**0**	**0**	**0**	1.18e-09	5.30e-09	**0**	**0**	**0**	**0**	**0**	**0**
F13	4.93e-04	2.20e-03	6.28e-11	2.71e-11	**0**	**0**	4.35e-14	7.51e-14	**0**	**0**	**0**	**0**
F14	-	-	-	-	-	-	-	-	-	-	2.72e-03	4.41e-03
F15	3.07e-14	2.88e-15	6.41e-14	3.17e-15	1.93e-14	4.07e-15	3.19e-14	6.15e-15	2.84e-14	5.02e-15	**1.59e-14**	**3.19e-15**
F16	**1.57e-32**	**2.80e-48**	1.59e-32	5.77e-34	**1.57e-32**	**2.80e-48**	**1.57e-32**	**0**	**1.57e-32**	**5.57e-48**	**1.57e-32**	**5.57e-48**
F17	**1.34e-32**	**2.80e-48**	2.97e-32	1.75e-32	**1.34 e-32**	**2.80e-48**	**1.34 e-32**	**0**	**1.34e-32**	**5.57e-48**	**1.34e-32**	**5.57e-48**
F18	3.53e-19	2.66e-19	7.02e-17	4.23e-17	7.22e-17	1.90e-16	7.40e-28	1.28e-28	**1.30e-29**	**4.89e-29**	1.90e-29	4.85e-29
F19	**1.34e-31**	**0**	6.20e-31	4.64e-31	**1.34e-31**	**0**	**1.34e-31**	**0**	**1.34e-31**	**6.68e-47**	**1.34e-31**	**6.68e-47**
F20	**0**	**0**	1.42e-15	3.17e-15	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
F21	**-7.83e+01**	**1.63e-09**	**-78.3323**	**4.50e-08**	**-7.83e+01**	**8.60e-11**	**-78.3323**	**1.06e-13**	**-7.83e+01**	**3.80e-13**	**-78.3323**	**2.64e-13**
F22	-9.56e+01	4.24e-01	-93.4167	6.48e-01	-9.52e+01	6.15e-01	**-98.6923**	**1.24e-01**	-9.85e+01	5.68e-01	-97.22	3.11e-01

Tableau III.6 : Comparaison des performances des algorithmes mCLABC, ABCVSS, MuABC, OGABC, ABCbest et CABC sur des problèmes d'optimisation des fonctions F1-F20 (D = 60) et des fonctions F21-F22 (D = 200).

Fonction	CABC		ABCbest		OGABC		MuABC		ABCVSS		mCLABC	
	Moy.	Éc.-Type	Moy.	Éc.-Type	Moy.	Éc.-Type	Moy.	Éc.-Type	Moy.	Éc.-Type	Moy.	Éc.-Type
F1	2.63e-34	2.68e-34	1.27e-29	9.42e-30	7.99e-44	9.34e-44	5.73e-49	8.10e-49	5.24e-53	2.86e-52	**2.68e-53**	**3.21e-53**
F2	3.43e-30	3.60e-30	5.68e-26	5.14e-26	7.09e-40	6.85e-40	2.76e-46	1.31e-46	**1.68e-51**	**7.53e-51**	8.50e-50	1.25e-49
F3	1.64e-34	1.45e-34	4.39e-30	4.08e-30	6.02e-45	5.84e-45	7.36e-50	5.88e-50	4.06e-53	1.58e-52	**4.50e-54**	**4.47e-54**
F4	2.65e-47	5.54e-47	1.36e-58	2.75e-58	3.05e-70	3.43e-70	2.47e-68	2.56e-68	8.18e-74	3.55e-73	**4.57e-108**	**2.25e-107**
F5	5.19e-18	2.10e-18	5.26e-16	1.67e-16	7.64e-23	9.40e-24	7.73e-27	4.20e-27	1.76e-27	7.06e-27	**1.00e-29**	**1.56e-29**
F6	3.99e+01	7.62e+00	3.28e+01	3.64e+00	**5.19e+00**	**1.37e+00**	3.13e+01	2.14e+00	6.01e+00	1.04e+00	7.19e+00	1.55e+00
F7	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
F8	-	-	-	-	-	-	-	-	-	-	9.11e-102	3.49e-101
F9	1.99e-01	6.07e-02	1.63e-01	2.93e-02	**7.64e-03**	**1.95e-03**	1.16e-01	5.88e-03	5.76e-02	1.16e-02	1.48 e-02	5.13e-03
F10	3.27e-01	6.37e-01	2.38+01	3.15+01	1.49e+00	1.27e+00	2.14e-02	2.19e-02	3.68e-01	8.48e-01	**1.87 e-02**	**5.72e-02**
F11	**0**	**0**	**0**	**0**	2.63e-13	1.31e-12	**0**	**0**	**0**	**0**	**0**	**0**
F12	**0**	**0**	**0**	**0**	9.78e-14	4.26e-13	**0**	**0**	**0**	**0**	**0**	**0**
F13	1.49e-08	3.31e-08	1.79e-07	3.83e-07	2.95e-04	1.47e-03	7.58e-15	9.77e-15	**0**	**0**	**0**	**0**
F14	-	-	-	-	-	-	-	-	-	-	2.59e-03	4.37e-03
F15	8.31e-14	6.92e-15	9.66e-14	5.94e-15	**7.81e-15**	**1.58e-15**	2.28e-14	3.10e-15	7.02e-14	6.44e-15	4.10e-14	6.34e-15
F16	**7.85e-33**	**2.79e-48**	8.50e-32	4.14e-32	**7.85e-33**	**2.79e-48**	**7.85e-33**	**0**	**7.85e-33**	**2.78e-48**	**7.85e-33**	**2.78e-48**
F17	**1.34e-32**	**5.58e-48**	1.68e-30	6.17e-31	**1.34e-32**	**5.58e-48**	**1.34e-32**	**0**	**1.34e-32**	**5.57e-48**	**1.34e-32**	**5.57e-48**
F18	4.51e-18	8.82e-18	3.61e-14	3.96e-15	3.95e-16	5.76e-16	7.40e-28	1.28e-29	3.49e-27	1.89e-26	**4.87e-28**	**7.57e-28**
F19	**1.34e-32**	**2.23e-47**	9.46e-30	1.15e-29	**1.34e-32**	**2.23e-47**	**1.34e-31**	**0**	**1.34e-31**	**6.68e-47**	**1.34e-31**	**6.68e-47**
F20	8.52e-15	7.78e-15	2.55e-14	6.35e-15	6.87e-05	1.53e-04	8.36e-16	8.20e-15	9.47e-16	3.61e-15	**0**	**0**
F21	**-7.83e+01**	**1.70e-09**	**-78.3323**	**8.28e-08**	**-7.83e+01**	**1.35e-09**	**-78.3323**	**1.42e-14**	**-7.83e+01**	**5.12e-14**	**-78.33**	**2.32e-12**
F22	-1.87e+02	8.33e-01	-181.1403	7.73e-01	-1.90e+02	8.73e-01	**-196.65**	**2.84e-01**	-1.96e+02	1.42e+00	-192.635	4.35e-01

Tableau III.7 : Analyse statistique de la performance de la stratégie mCLABC proposée sur les problèmes d'optimisation des fonctions numériques F1-F20 (D = 15) et F21-F22 (D = 50).

Fonction	CABC		ABCbest		MuABC		mCLABC	
	AVEN	SR	AVEN	SR	AVEN	SR	AVEN	SR
F1	1.56e+04	**100**	1.62e+04	**100**	1.11e+04	**100**	**9.60e+03**	**100**
F2	2.07e+04	**100**	2.05e+04	**100**	1.41e+04	**100**	**1.25e+04**	**100**
F3	1.42e+04	**100**	1.44e+04	**100**	1.02e+04	**100**	**8.52e+03**	**100**
F4	5.66e+03	**100**	4.87e+03	**100**	**4.14e+03**	**100**	5.32e+03	**100**
F5	2.41e+04	**100**	2.42e+04	**100**	1.70e+04	**100**	**1.41e+04**	**100**
F6	1.16e+04	**100**	3.09e+03	**100**	2.71e+03	**100**	**1.74e+03**	**100**
F7	6.02e+03	**100**	6.19e+03	**100**	4.10e+03	**100**	**3.85e+03**	**100**
F8	-	-	-	-	-	-	3.76e+03	**100**
F9	3.56e+04	**100**	4.93e+03	**100**	3.81e+03	**100**	**3.54e+03**	**100**
F10	1.42e+04	**100**	1.87e+04	70	1.03e+04	**100**	**9.57e+03**	**100**
F11	1.80e+04	**100**	1.87e+04	**100**	1.42e+04	**100**	**1.17e+04**	**100**
F12	1.90e+04	**100**	1.91e+04	**100**	1.49e+04	**100**	**1.23e+04**	**100**
F13	2.65e+04	96	3.14e+04	68	**2.24e+04**	**100**	2.57e+04	86.67
F14	-	-	-	-	-	-	7.48e+03	**100**
F15	2.58e+04	**100**	2.61e+04	**100**	1.83e+04	**100**	**1.53e+04**	**100**
F16	1.33e+04	**100**	1.40e+04	**100**	9.54e+03	**100**	**8.56e+03**	**100**
F17	1.47e+04	**100**	1.51e+04	**100**	1.07e+04	**100**	**9.09e+03**	**100**
F18	2.30e+04	**100**	2.52e+04	**100**	1.85e+04	**100**	**1.76e+04**	**100**
F19	1.69e+04	**100**	1.66e+04	**100**	1.08e+04	**100**	**1.05e+04**	**100**
F20	2.80e+04	**100**	2.86e+04	**100**	1.91e+04	**100**	**1.76e+04**	**100**
F21	1.43e+04	**100**	1.59e+04	**100**	1.18e+04	**100**	**1.13e+04**	**100**
F22	3.59e+04	82	-	0	**3.23e+04**	**100**	4.53e+04	**100**

Tableau III.8 : Analyse statistique de la performance de la stratégie mCLABC proposée sur les problèmes d'optimisation des fonctions numériques F1-F20 (D = 30) et F21-F22 (D = 100).

Fonction	CABC		ABCbest		MuABC		mCLABC	
	AVEN	SR	AVEN	SR	AVEN	SR	AVEN	SR
F1	3.30e+04	**100**	3.53e+04	**100**	2.38e+04	**100**	**1.99e+04**	**100**
F2	4.39e+04	**100**	4.61e+04	**100**	2.93e+04	**100**	**2.59e+04**	**100**
F3	3.09e+04	**100**	3.37e+04	**100**	2.17e+04	**100**	**1.86e+04**	**100**
F4	1.22e+04	**100**	9.80e+03	**100**	**5.62e+03**	**100**	1.41e+04	**100**
F5	5.07e+04	**100**	5.46e+04	**100**	3.44e+04	**100**	**2.99e+04**	**100**
F6	5.32e+04	**100**	1.67e+04	**100**	1.39e+04	**100**	**1.18e+04**	**100**
F7	1.34e+04	**100**	1.46e+04	**100**	8.90e+03	**100**	**8.76e+03**	**100**
F8	-	-	-	-	-	-	8.32e+03	**100**
F9	3.63e+04	**100**	3.30e+04	**100**	**1.77e+04**	**100**	1.80e+04	**100**
F10	3.52e+04	**100**	5.51e+04	60	**2.27e+04**	**100**	2.54e+04	**100**
F11	4.15e+04	**100**	4.34e+04	**100**	**3.92e+04**	**100**	5.56e+04	**100**
F12	4.24e+04	**100**	4.28e+04	**100**	3.25e+04	**100**	**2.60e+04**	**100**
F13	4.45e+04	96	4.26e+04	**100**	**3.15e+04**	**100**	4.49e+04	**100**
F14	-	-	-	-	-	-	1.60e+04	**100**
F15	5.32e+04	**100**	5.75e+04	**100**	3.72e+04	**100**	**3.16e+04**	**100**
F16	2.78e+04	**100**	3.05e+04	**100**	1.91e+04	**100**	**1.74e+04**	**100**
F17	3.11e+04	**100**	3.36e+04	**100**	2.21e+04	**100**	**1.91e+04**	**100**
F18	5.03e+04	**100**	5.61e+04	**100**	3.91e+04	**100**	**3.85e+04**	**100**
F19	3.51e+04	**100**	2.46e+04	**100**	2.26e+04	**100**	**2.14e+04**	**100**
F20	5.83e+04	**100**	6.20e+04	**100**	**3.93e+04**	**100**	4.02e+04	**100**
F21	3.09e+04	**100**	4.50e+04	**100**	**2.35e+04**	**100**	2.36e+04	**100**
F22	9.45e+04	82	-	0	**5.43e+04**	**100**	6.87e+04	**100**

III.3.2 Problèmes d'optimisation de fonctions complexes

Dans cette deuxième phase d'analyse de performance de la stratégie d'optimisation proposée, nous examinons à présent l'application du modèle mCLABC à une variété de problèmes d'optimisation de fonctions complexes catégorisées sous les labels CEC'05 et

CEC'13. Ces fonctions présentent des caractéristiques différentes de multimodalité, de séparabilité, de rotation et de discontinuité (voir paragraphe II.6.1 du chapitre II).

Tableau III.9 : Analyse statistique de la performance de la stratégie mCLABC proposée sur les problèmes d'optimisation des fonctions numériques F1-F20 (D = 60) et F21-F22 (D = 200).

Fonction	CABC		ABCbest		MuABC		mCLABC	
	AVEN	SR	AVEN	SR	AVEN	SR	AVEN	SR
F1	6.96e+04	**100**	7.87e+04	**100**	4.92e+04	**100**	**4.16e+04**	**100**
F2	9.06e+04	**100**	1.01e+05	**100**	6.07e+04	**100**	**5.51e+04**	**100**
F3	6.66e+04	**100**	7.62e+04	**100**	4.65e+04	**100**	**3.99e+04**	**100**
F4	2.46e+04	**100**	2.11e+04	**100**	**1.19e+04**	**100**	3.82e+04	**100**
F5	1.06e+05	**100**	1.18e+05	**100**	7.21e+05	**100**	**6.28e+04**	**100**
F6	1.52e+05	62	1.26e+05	**100**	1.12e+05	**100**	**4.45e+04**	**100**
F7	2.83e+04	**100**	3.25e+04	**100**	1.93e+04	**100**	**1.82e+04**	**100**
F8	-	-	-	-	-	-	1.84e+04	**100**
F9	-	0	-	0	-	0	**4.75e+04**	**100**
F10	8.70e+04	**100**	1.71e+05	22	**5.47e+04**	**100**	7.55e+04	**100**
F11	8.20e+04	**100**	9.13e+05	**100**	**7.65e+04**	**100**	1.26e+05	**100**
F12	8.56e+04	**100**	9.47e+04	**100**	724e+04	**100**	**5.56e+04**	**100**
F13	8.64e+04	86	8.99e+04	**100**	6.29e+04	**100**	**1.03e+04**	**100**
F14	-	-	-	-	-	-	3.44e+04	**100**
F15	1.10e+05	**100**	1.21e+05	**100**	7.56e+04	**100**	**6.62e+04**	**100**
F16	5.62e+04	**100**	6.46e+04	**100**	3.93e+04	**100**	**3.60e+04**	**100**
F17	6.50e+04	**100**	7.46e+04	**100**	4.39e+04	**100**	**3.96e+04**	**100**
F18	1.03e+05	**100**	1.26e+05	**100**	**7.94e+04**	**100**	8.36e+04	**100**
F19	7.42e+04	**100**	7.47e+04	**100**	4.46e+04	**100**	**4.34e+04**	**100**
F20	1.20e+05	**100**	1.09e+05	**100**	**8.14e+04**	**100**	8.69e+04	**100**
F21	6.38e+04	**100**	7.26e+04	**100**	**4.69e+04**	**100**	4.96e+04	**100**
F22	-	0	-	0	**1.16e+05**	**100**	1.55e+05	**100**

III.3.2.1 Application aux fonctions composites CEC'05

Deux tests comparatifs sont établis : le premier test compare le modèle proposé à des variantes récentes du modèle ABC, le deuxième dresse une étude comparative par rapport à des variantes avancées de la stratégie d'optimisation à évolution différentielle (DE).

III.3.2.1.1 Résultats comparatifs des variantes du modèle ABC

Dans un premier test, la stratégie d'optimisation mCLABC est appliquée aux fonctions complexes CEC'05 en considérant des problèmes de dimension 10. Le modèle mCLABC proposé est comparé à des variantes avancées du modèle ABC. Pour l'ensemble des cas étudiés, la taille de population choisie est 40, le nombre maximal d'évaluations de la fonction objectif (Max_FEs) est fixé à 320.000 et le paramètre $limit$ est de 100. Les résultats comparatifs des modèles mCLABC, GB-ABC (Binary version ABC based on genetic operators) [85], QBPSO (Quantum inspired binary PSO) [86] et DisABC (Binary version of the ABC) [87] sont présentés dans le tableau III.10. À partir de ce tableau, il est facile de remarquer que la stratégie mCLABC achève les meilleures performances par rapport aux algorithmes comparés, et ce pour les quatre fonctions unimodales complexes f1, f2, f4 et f5. Il

est important de noter que le modèle mCLABC proposé est le seul algorithme capable d'obtenir une solution optimale globale pour la fonction f1 (résoudre parfaitement f1) et une solution optimale quasi-globale pour les fonctions f4 et f5. Pour les fonctions multimodales f6-f17 qui sont à caractéristiques difficiles, nous pouvons constater que l'algorithme mCLABC surpasse les approches DisABC, QBPSO et GB-ABC pour les fonctions f6, f7, f9 et f11- f17. L'algorithme mCLABC était capable de résoudre parfaitement (atteindre l'optimum global) les deux fonctions multimodales f9 et f15. Ainsi, aucun des algorithmes comparés n'a pu atteindre la solution optimale globale ou quasi-optimale, sauf le modèle mCLABC proposé.

Tableau III.10 : Comparaison des performances du modèle mCLABC sur les fonctions composites CEC'05 de dimension 10.

Fonction	DisABC		QBPSO		GB-ABC		mCLABC	
	Moy.	Éc.-Type	Moy.	Éc.-Type	Moy.	Éc.-Type	Moy.	Éc.-Type
f1	-103.56	142.39	-334.94	78.78	-336.54	112.75	**-450**	**0**
f2	-74.43	170.08	-270.45	173.75	-280.57	151.46	**-450**	**2.35e-28**
f3	-44.47	161.15	-291.45	153.19	**-303.56**	**145.54**	7080.96	22996.50
f4	-64.12	154.92	-332.52	122.57	-341.57	96.855	**-450**	**1.39e-28**
f5	83.19	155.51	-115.02	166.90	-136.87	177.21	**-309.99**	**7.32e-13**
f6	740.04	183.25	577.09	138.99	533.92	127.21	**390**	**4.17e-25**
f7	-149.06	15.47	-171.42	8.82	-167.25	9.06	**-179.84**	**0.163**
f8	-119.66	0.08	**-119.79**	**0.07**	-119.73	0.09	-119.75	0.057
f9	-309.69	0.07	-309.81	0.06	-309.67	0.11	**-330**	**0**
f10	-309.67	0.07	**-309.78**	**0.08**	-309.66	0.12	102.64	5.896
f11	110.29	0.09	110.19	0.09	110.32	0.10	**93.21**	**0.999**
f12	-439.67	0.08	-439.81	0.07	-439.75	0.09	**-456.33**	**7.15**
f13	-109.68	0.08	-109.81	0.09	-109.68	0.12	**-129.92**	**0.074**
f14	-279.66	0.08	-279.75	0.08	-279.64	0.09	**-297.50**	**0.437**
f15	472.64	48.95	435.10	174.28	412.57	189.24	**120**	**0**
f16	472.43	67.26	449.07	160.37	411.04	164.88	**240.195**	**11.99**
f17	472.22	59.78	398.99	183.57	367.16	165.15	**238.45**	**10.99**

III.3.2.1.2 Résultats comparatifs des variantes du modèle DE

Dans ce deuxième test comparatif, l'algorithme mCLABC est comparé à des variantes récentes de la stratégie différentielle DE. Une série de tests est menée sur 25 fonctions CEC'05 définissant des problèmes de dimension $D = 30$. Le critère d'arrêt est fixé à 300.000 évaluations de la fonction objectif (Max_FEs), la taille de la population est 60, et la valeur $limit = 900$. Les résultats obtenus sont dressés dans le tableau III.11. Le paramétrage des algorithmes comparés et les résultats qui leur sont associés sont repris des travaux publiés dans [88].

Les variantes de la stratégie DE considérés dans l'étude comparative renferment des modèles d'optimisation très puissants dont l'algorithme jDE (DE algorithm based on self-adaptive parameter control) [38], l'algorithme SaDE (Self adaptive DE) [39], l'algorithme

CoDE (Composite DE) [89], et l'algorithme CoBiDE (DE based on covariance matrix learning and bimodal distribution parameter setting) [88].

Tableau III.11 : Comparaison des performances du modèle mCLABC sur les fonctions composites CEC'05 de dimension 30.

Fonction	jDE		SaDE		CoDE		CoBiDE		mCLABC	
	Moy.	Éc.-Type	Moy.	Éc.-Type	Moy.	Éc.-Type	Moy.	Éc.-Type	Moy.	Éc.-Type
f1	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
f2	1.11e-06	1.96e-06	8.26e-06	1.65e-05	**1.69e-15**	**3.95e-15**	1.60e-12	2.90e-12	1.74e-13	3.37e-13
f3	1.98e+05	1.10e+05	4.27e+05	2.08e+05	1.05e+05	6.25e+04	**7.26e+04**	**5.64e+04**	1.95e+05	1.38e+05
f4	4.40e-02	1.26e-01	1.77e+02	2.67e+02	5.81e-03	1.38e-02	**1.16e-03**	**2.74e-03**	6.89e+02	1.14e+03
f5	5.11e+02	4.40e+02	3.25e+03	5.90e+02	3.31e+02	3.44e+02	**8.03e+01**	**1.51e+02**	3.07e+03	6.17e+02
f6	2.35e+01	2.50e+01	5.31e+01	3.25e+01	1.60e-01	7.85e-01	4.13e-02	9.21e-02	**9.04e-08**	**2.55e-07**
f7	1.18e−02	7.78e-03	1.57e-02	1.38e-02	7.46e-03	8.55e-03	**1.77e-03**	**3.73e-03**	2.37e-02	1.48e-02
f8	2.09e+01	4.86e-02	2.09e+01	4.95e-02	**2.01e+01**	1.41e-01	2.07e+01	3.75e-01	2.09e+01	6.69e-02
f9	**0**	**0**	2.39e-01	4.33e-01	**0**	**0**	**0**	**0**	**0**	**0**
f10	5.54e+01	8.46e+00	4.72e+01	1.01e+01	**4.15e+01**	**1.16e+01**	4.41e+01	1.29e+01	1.04e+02	4.47e+01
f11	2.79e+01	1.61e+00	1.65e+01	2.42e+00	1.18e+01	3.40e+00	**5.62e+00**	**2.19e+00**	2.36e+01	3.03e+00
f12	8.63e+03	8.31e+03	3.02e+03	2.33e+03	3.05e+03	3.80e+03	2.94e+03	3.93e+03	**2.41e+03**	**2.55e+03**
f13	1.66e+00	1.35e-01	3.94e+00	2.81e-01	1.57e+00	3.27e-01	2.64e+00	1.13e+00	**8.56e-01**	**1.60e-01**
f14	1.30e+01	2.00e-01	1.26e+01	2.83e-01	1.23e+01	4.81e-01	1.23e+01	4.90e-01	**1.23e+01**	**3.18e-01**
f15	3.77e+02	8.02e+01	3.76e+02	7.83e+01	3.88e+02	6.85e+01	4.04e+02	5.03e+01	**1.96e+01**	**3.66e+01**
f16	7.94e+01	2.96e+01	8.57e+01	6.94e+01	7.37e+01	5.13e+01	**7.38e+01**	**3.66e+01**	2.02e+02	4.19e+01
f17	1.37e+02	3.80e+01	7.83e+01	3.76e+01	**6.67e+01**	**2.12e+01**	7.25e+01	2.02e+01	1.18e+02	3.79e+01
f18	9.04e+02	1.08e+01	8.68e+02	6.23e+01	9.04e+02	1.04e+00	9.03e+02	1.05e+01	**8.52e+02**	**6.43e+00**
f19	9.04e+02	1.11e+00	8.74e+02	6.22e+01	9.04e+02	9.42e-01	9.03e+02	1.04e+01	**8.51e+02**	**6.12e+00**
f20	9.04e+02	1.10e+00	8.78e+02	6.03e+01	9.04e+02	9.01e-01	9.04e+02	5.95e-01	**8.50e+02**	**1.15e+01**
f21	**5.00e+02**	**4.80e-13**	5.52e+02	1.82e+02	**5.00e+02**	**4.88e-13**	**5.00e+02**	**4.62e-13**	**5.00e+02**	**4.98e-13**
f22	8.75e+02	1.91e+01	9.36e+02	**1.83e+01**	8.63e+02	2.43e+01	8.62e+02	2.80e+01	**7.02e+02**	1.66e+02
f23	**5.34e+02**	**2.77e-04**	**5.34e+02**	**3.57e-03**	**5.34e+02**	**4.12e-04**	**5.34e+02**	**1.30e-04**	**5.34e+02**	**3.20e-04**
f24	**2.00e+02**	**2.85e-14**	**2.00e+02**	**6.20e-13**	**2.00e+02**	**2.85e-14**	**2.00e+02**	**2.85e-14**	**2.00e+02**	**1.17e-13**
f25	2.11e+02	**7.32e-01**	2.14e+02	2.00e+00	2.11e+02	9.02e-01	2.10e+02	7.73e-01	**2.03e+02**	8.89e+00

Sur la base des résultats résumés dans le tableau III.11, il y a lieu de conclure que le modèle mCLABC proposé achève des résultats équivalents ou performants par rapport à quatre variantes du modèle DE sur la majorité des fonctions complexes testées à l'exemple des fonctions multimodales de base f6-f12, des fonctions multimodales étendues f13-f14 et des fonctions de composition hybride f15, f18-f20, f22 et f25. Ces résultats sont équivalents ou meilleurs à ceux de jDE sur 20 tests, SaDE sur 19 tests, CoDE sur 15 tests et CoBiDE sur 16 tests. D'autre part, une performance remarquable de la stratégie mCLABC proposée peut être constatée sur la fonction multimodale f6 qui affiche une erreur moyenne de 9.04e-08 et une déviation de 2.55e-07 obtenus sur 25 essais indépendants. Enfin, le nouveau modèle mCLABC est aussi performant qu'au moins une variante de la stratégie DE sur cinq autres fonctions testées.

III.3.2.2 Application aux fonctions composites CEC'13

Dans cette phase d'étude, nous considérons des problèmes plus complexes et de grande dimension. Il s'agit des fonctions composites de type CEC'13, tout récemment catégorisées.

Les tests comparatifs sont menés sur quatre variantes du modèle ABC: PABC (ABC with Powell's local search) [90], MABC (Modified ABC) [26], comABC (Compact ABC) [59], et EcABC (Enhanced compact ABC) [91], ainsi que sur deux variantes de la stratégie différentielle DE: cDE (Persistent compact DE) [92], et McDE (Memetic compact DE) [93]. Les tests sont réalisés sur 28 fonctions benchmark de dimensions D = 50 et D = 100. Pour assurer une comparaison fiable, les mêmes paramètres de contrôle utilisés dans [92] pour les méthodes comparées sont repris dans le modèle mCLABC. Le nombre d'évaluations de la fonction objectif est fixé à 5.000*D. Les résultats consignés dans les tableaux III.12 et III.13 correspondent aux valeurs moyennes (Moy.) obtenues sur 30 essais pour chacun des algorithmes testés, et aux déviations (Éc.-Type) associées.

Tableau III.12 : Comparaison des performances du modèle mCLABC sur les fonctions composites CEC'13 de dimension 50.

Fonc.	PABC		MABC		comABC		EcABC		mCLABC	
	Moy.	Éc.-Type	Moy.	Éc.-Type	Moy.	Éc.-Type	Moy.	Éc.-Type	Moy.	Éc.-Type
f1	1.364e-12	1.49e-13	1.606e-12	9.18e-13	1.062e-01	6.65e-01	1.591e-12	1.84e-13	**2.274e-13**	**1.45e-13**
f2	1.299e+07	9.72e+06	2.080e+07	6.04e+06	5.645e+08	4.10e+07	**5.832e+06**	**3.11e+05**	5.784e+06	4.02e+06
f3	1.474e+09	8.25e+08	1.083e+09	2.13e+09	2.899e+11	9.53e+10	6.312e+08	4.35e+07	**1.561e+08**	**2.31e+08**
f4	1.443e+05	4.32e+04	1.578e+05	1.94e+04	7.308e+04	8.28e+03	**3.398e+04**	**2.25e+03**	1.150e+05	1.89e+04
f5	1.250e-12	8.31e-13	1.477e-12	8.23e-12	1.107e+03	3.64e+02	6.063e-04	2.30e-05	**1.137e-13**	**3.77e-13**
f6	4.408e+02	3.99e+01	4.674e+01	7.01e+00	2.244e+03	7.36e+02	**3.968e+01**	**9.39e-01**	4.285e+01	5.07e-01
f7	1.636e+02	6.56e+01	1.498e+02	5.16e+01	1.475e+02	8.05e+02	1.856e+02	6.22e+01	**1.056e+02**	**1.46e+01**
f8	2.117e+01	9.84e+00	2.123e+01	2.46e+00	2.114e+01	5.21e+00	**2.106e+01**	**3.14e+00**	**2.115e+01**	**3.58e-02**
f9	6.013e+01	1.06e+00	6.156e+01	4.87e+00	7.791e+01	2.23e+01	7.141e+01	1.44e+01	**5.663e+01**	**4.05e+00**
f10	2.338e+00	9.28e-02	2.713e+00	4.92e-01	2.663e+12	4.58e+02	9.705e-02	4.46e-03	**8.630e-02**	**5.80e-02**
f11	2.273e-13	1.98e-14	4.705e-13	8.46e-14	7.757e+02	5.73e+01	2.855e+02	9.63e+00	**5.684e-14**	**3.04e-14**
f12	3.841e+02	9.51e+01	8.257e+02	1.74e+02	8.840e+02	6.48e+01	8.327e+02	7.35e+01	**1.959e+02**	**5.10e+01**
f13	6.861e+02	4.22e+01	7.038e+02	4.18e+01	8.049e+02	6.12e+01	7.561e+02	8.01e+01	**3.023e+02**	**4.45e+01**
f14	1.165e+01	4.17e+00	**3.409e+00**	**3.91e-01**	1.057e+04	1.18e+04	3.733e+03	1.07e+02	**2.124e+00**	**2.31e+00**
f15	8.880e+03	6.01e+02	**6.489e+03**	**2.56e+02**	1.404e+04	7.85e+03	6.504e+03	9.17e+02	8.241e+03	6.90e+02
f16	1.967e+00	5.39e+00	1.524e+00	7.92e-01	3.648e+00	5.62e+00	**1.201e+00**	**6.02e-01**	2.194e+00	3.44e-01
f17	5.150e+01	1.71e+00	5.104e+01	3.07e+01	1.025e+03	2.03e+03	5.982e+02	4.08e+01	**5.079e+01**	**1.37e-02**
f18	5.612e+02	4.04e+01	7.895e+02	2.42e+01	1.237e+03	8.91e+02	1.381e+03	4.38e+02	**2.223e+02**	**3.24e+01**
f19	2.272e+00	1.75e-01	9.302e+01	3.76e+02	5.088e+05	3.48e+01	2.504e+01	5.21e+00	**7.095e-01**	**1.99e-01**
f20	8.665e+01	8.81e+00	2.457e+01	1.51e+00	2.503e+01	3.73e+00	**2.051e+01**	**6.24e+00**	**2.217e+01**	**8.89e-01**
f21	**2.002e+02**	**3.26e+01**	2.017e+03	9.87e+02	4.509e+03	2.84e+03	1.122e+03	3.67e+02	3.883e+02	3.28e+02
f22	**5.996e+01**	**1.72e+00**	2.714e+03	6.23e+02	1.030e+04	7.32e+03	2.562e+03	8.85e+02	4.098e+01	3.01e+01
f23	1.018e+04	4.65e+03	8.459e+04	6.86e+03	1.552e+04	5.08e+03	9.586e+03	2.61e+02	**2.316e+03**	**9.70e+02**
f24	3.451e+02	1.06e+01	3.865e+02	2.32e+01	4.453e+02	4.19e+01	3.787e+02	1.36e+01	**3.315e+02**	**1.85e+01**
f25	3.763e+02	8.51e+01	4.257e+02	1.58e+01	4.192e+02	1.74e+02	3.866e+02	9.87e+00	**3.570e+02**	**5.64e+00**
f26	2.015e+02	3.56e+01	2.022e+02	3.56e+01	5.083e+02	3.67e+01	2.076e+02	1.97e+01	**2.003e+02**	**1.05e-01**
f27	1.946e+03	6.98e+02	2.207e+03	1.85e+03	2.524e+03	2.44e+02	**1.892e+03**	**8.90e+01**	**1.584e+03**	**4.59e+02**
f28	**3.782e+02**	**3.58e+01**	4.109e+03	2.81e+02	7.632e+03	2.86e+03	2.116e+03	7.79e+02	**4.000e+02**	**1.16e-12**

Comme il peut être constaté à partir des tableaux III.12 et III.13, les résultats du modèle mCLABC sont concluants sur 21 fonctions composites de dimension D = 50, et 23 fonctions de dimension D = 100. La supériorité du modèle mCLABC est clairement visible pour les fonctions unimodales f1, f3 et f5, il se place cependant en deuxième rang par rapport au modèle EcABC pour les cas f2 et f4. Néanmoins, pour ces deux derniers cas, seul le modèle mCLABC arrive à trouver le quasi-optimum de f1 et f5 pour une dimension 100.

Pour l'ensemble des fonctions de composition multimodales f6-f20, EcABC obtient des résultats supérieurs sur f6 et f16 (avec D = 50), alors que MABC et McDE présentent de meilleures performances sur f15 avec D = 50 et D = 100, respectivement. Pour le reste des fonctions composites multimodales testées, mCLABC conclut des performances visiblement supérieures.

Tableau III.13 : Comparaison des performances du modèle mCLABC sur les fonctions composites CEC'13 de dimension 100.

Fonction	cDE		McDE		comABC		EcABC		mCLABC	
	Moy.	Éc.-Type	Moy.	Éc.-Type	Moy.	Éc.-Type	Moy.	Éc.-Type	Moy.	Éc.-Type
f1	4.063e-02	8.14e-03	1.077e+05	9.02e+04	3.150e+00	4.17e-01	1.294e-02	2.78e-03	**2.273e-13**	**4.97e-13**
f2	1.986e+08	9.13e+07	1.169e+09	4.89e+08	1.840e+09	2.93e+08	**5.617e+06**	**9.57e+06**	1.337e+07	5.80e+06
f3	5.179e+10	3.71e+10	1.790e+16	3.37e+16	4.740e+14	3.69e+13	3.058e+09	1.57e+09	**1.538e+09**	**1.20e+09**
f4	1.071e+05	7.15e+04	1.879e+05	2.41e+04	2.548e+05	4.03e+04	**7.721e+03**	**4.85e+04**	2.171e+05	3.27e+04
f5	4.209e-02	9.33e-02	1.336e+04	9.56e+03	5.688e-01	5.75e-01	1.755e-02	9.59e-01	**5.684e-13**	**8.15e-13**
f6	5.598e+03	3.92e+02	2.735e+04	1.68e+04	1.436e+04	6.49e+05	**1.548e+02**	**6.78e+02**	**1.886e+02**	**2.33e+01**
f7	8.156e+04	7.06e+03	1.452e+04	7.31e+03	3.175e+06	4.51e+06	3.893e+05	2.76e+05	**1.407e+02**	**2.11e+01**
f8	2.129e+01	6.94e+00	3.311e+01	6.44e+00	2.323e+01	2.96e+00	2.132e+01	8.23e+00	**2.129e+01**	**3.32e-02**
f9	1.397e+02	4.38e+02	1.605e+02	3.68e+01	4.640e+02	1.83e+01	1.495e+02	7.95e+01	**1.335e+02**	**5.58e+00**
f10	2.786e+03	6.46e+02	7.107e+03	6.25e+03	1.356e+04	7.85e+03	5.998e+02	1.86e+02	**8.44e-02**	**6.58e-02**
f11	1.021e+03	6.79e+02	9.425e+02	7.75e+02	8.959e+02	4.86e+02	2.147e+02	7.09e+02	**5.684e-14**	**1.32e-13**
f12	1.239e+03	4.98e+03	2.063e+03	5.08e+02	3.762e+03	3.06e+02	1.994e+03	6.55e+01	**5.978e+02**	**7.07e+01**
f13	1.921e+03	2.55e+02	2.267e+03	7.94e+02	2.685e+03	8.71e+03	2.713e+03	5.85e+02	**8.179e+02**	**8.10e+01**
f14	9.034e+03	5.05e+03	1.014e+04	8.01e+03	1.391e+04	2.78e+04	1.051e+03	5.47e+02	**3.532e+00**	**1.59e+00**
f15	1.746e+04	8.48e+03	**1.126e+04**	**3.52e+03**	2.058e+04	9.39e+03	1.934e+04	1.38e+02	1.701e+04	1.19e+03
f16	3.106e+00	2.54e+00	3.636e+00	8.75e-01	4.351e+00	5.54e-01	**1.797e+00**	**9.29e-01**	2.696e+00	3.17e-01
f17	1.221e+03	6.16e+02	2.561e+03	3.01e+02	2.404e+03	2.07e+02	1.064e+03	3.49e+01	**1.016e+02**	**4.06e-02**
f18	3.085e+03	4.73e+03	2.389e+03	8.41e+02	2.558e+03	2.31e+03	2.721e+03	5.85e+02	**5.789 e+02**	**7.13e+01**
f19	3.006e+02	5.49e+01	1.253e+05	1.75e+04	6.744e+05	2.25e+04	8.191e+01	7.57e+00	**1.898e+00**	**3.89e-01**
f20	**5.00e+01**	**0.00e+00**	**5.00e+01**	**0.00e+00**	**5.00e+01**	**0.00e+00**	**5.00e+01**	**0.00e+00**	**5.00e+01**	**0.00e+00**
f21	4.267e+02	9.34e+01	7.739e+03	1.84e+02	3.067e+03	4.21e+02	4.261e+02	7.79e+01	**3.820e+02**	**3.88e+01**
f22	1.095e+04	4.69e+04	1.346e+04	4.38e+03	1.777e+04	9.04e+03	2.082e+04	7.94e+03	**9.780e+01**	**4.75e+01**
f23	**2.279e+04**	**3.11e+03**	3.345e+04	2.58e+03	3.442e+04	4.08e+03	2.214e+04	6.01e+03	**2.308e+04**	**1.66e+03**
f24	7.651e+02	6.89e+01	7.355e+02	6.02e+02	6.791e+02	7.11e+01	6.163e+02	7.48e+00	**5.274e+02**	**3.43e+01**
f25	7.820e+02	4.50e+02	6.107e+02	3.21e+01	6.476e+02	1.17e+01	6.332e+02	1.52e+01	**5.574e+02**	**1.12e+01**
f26	6.624e+02	9.61e+01	7.052e+02	5.07e+01	7.445e+02	4.24e+01	2.076e+02	2.59e+01	**2.009e+02**	**2.72e-01**
f27	2.521e+03	4.31e+02	4.542e+03	2.62e+02	5.128e+03	8.01e+02	**3.204e+03**	**2.63e+02**	3.571e+03	8.43e+02
f28	1.508e+04	3.49e+03	1.446e+04	5.78e+03	1.815e+04	4.83e+03	1.417e+04	5.79e+03	**3.222e+03**	**7.98e+02**

Par ailleurs, pour les fonctions f21-f28, PABC surpasse mCLABC sur les fonctions f21 et f22 avec D = 50, alors que EcABC devance mCLABC sur seulement le cas f27 avec D = 100. Pour les autres cas de fonctions complexes, mCLABC réalise des résultats comparatifs équivalents ou supérieurs.

En somme, la performance de la stratégie mCLABC proposée se voit très clairement démontrée à travers les tests menés sur les fonctions composites CEC. Dans la plupart des cas considérés, le modèle proposé arrive à réaliser une solution optimale ou quasi-optimale très acceptable par rapport aux métaheuristiques comparées.

III.4 Analyse de robustesse de la stratégie d'optimisation proposée

Les conclusions que nous avions pu établir quant à la performance de la stratégie mCLABC proposée sont très concluantes pour l'ensemble des tests comparatifs menés sur les problèmes d'optimisation de fonctions standards et composites. L'analyse de performance présentée dans le paragraphe III.3 est fondée sur un paramétrage approprié de la structure algorithmique du modèle d'optimisation développé, dont la mise au point aurait découlée d'une analyse de robustesse que nous proposons de retracer dans la suite.

L'analyse de robustesse du modèle mCLABC est établie par rapport aux paramètres ajustables associés aux processus suivants :

- Le partitionnement des essaims d'abeilles Employed et Onlooker.
- L'altération des sources de nourriture lors du fourragement.
- Le mécanisme de recherche à règles multiples.

Pour étudier l'impact de chacun de ces paramètres, cinq problèmes d'optimisation de fonctions standards sont considérés. Il s'agit des fonctions F1, F10, F11, F13 et F15 avec une dimension D = 30. Les essais sont répétés 30 fois, avec un nombre maximum d'évaluations de la fonction objectif fixé à 100.000.

III.4.1 Impact du partitionnement des essaims d'abeilles

La stratégie mCLABC proposée emploie trois sous-groupes d'abeilles associés séparément à chaque catégorie d'abeilles Employed et Onlooker. Pour examiner l'impact du mécanisme de partitionnement en sous-groupes des essaims d'abeilles sur le modèle mCLABC, cinq schémas différents sont introduits. Par souci de clarté, on désigne un schéma donné par «mCLABC(EBS, OBS)» pour décrire un modèle d'optimisation mCLABC avec un nombre EBS de sous-groupes associés à la phase Employed et un nombre OBS de sous-groupes associés à la phase Onlooker. Les schémas simulés sont : mCLABC(3,1), mCLABC(1,3), mCLABC(2,2), mCLABC(4,4) et mCLABC(6,6).

Les résultats obtenus sont résumés dans le tableau III.14. Le modèle mCLABC proposé manifeste un caractère de recherche indépendant du nombre de sous-groupes associés aux essaims d'abeilles, ce qui traduit sa robustesse vis-à-vis du partitionnement des abeilles

Employed et Onlooker. Cependant, quelques petites constations sont à faire et qui n'influent en rien la validité de la conclusion induite. En effet, le modèle mCLABC(3,3) affiche une supériorité par rapport aux modèles mCLABC(3,1) et mCLABC(6,6) sur les fonctions multimodales F11 et F13, respectivement. Il est aussi performant que les modèles mCLABC(1,3), mCLABC(2,2) et mCLABC(4,4) pour ces mêmes fonctions. En outre, mCLABC(6,6) surpasse mCLABC pour les fonctions F10 et F15, mais converge vers un optimum local pour le cas F13.

Tableau III.14 : Analyse de robustesse du modèle mCLABC vis-à-vis du partitionnement des essaims d'abeilles.

Fonction	mCLABC (3,1)		mCLABC (1,3)		mCLABC (2,2)		mCLABC (4,4)		mCLABC (6,6)		mCLABC	
	Moy.	Éc.-Type	Moy.	Éc.-Type	Moy.	Éc.-Type	Moy.	Éc.-Type	Moy.	Éc.-Type	Moy.	Éc.-Type
F1	1.50e-55	2.88e-55	1.06e-50	2.81e-50	6.38e-54	1.08e-53	9.54e-57	2.15e-56	8.77e-57	1.11e-56	**8.08e-58**	**1.14e-57**
F10	2.96e-02	6.02e-02	1.37e-01	7.40e-01	1.66e-01	7.25e-01	7.37e-03	3.02 e-02	**4.91e-03**	**2.06e-02**	5.88e-03	1.66e-02
F11	1.18e-16	6.49e-16	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
F13	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	3.29e-04	1.80e-03	**0**	**0**
F15	1.81e-14	3.97e-15	1.81e-14	4.18e-15	2.24e-14	5.35e-15	1.47e-14	3.53e-15	**1.49e-14**	**2.08e-15**	1.59e-14	3.19e-15

III.4.2 Impact du processus d'altération des sources de nourriture

Lors du fourragement, les abeilles Employed et les abeilles Onlooker s'attèlent à modifier les caractéristiques des sources de nourriture en cours d'exploitation. Ainsi, le nombre de caractéristiques altérées en un temps de traitement donné pourrait affecter considérablement le processus d'exploitation des sources, ce qui impacterait la qualité des solutions. Pour analyser cet aspect, quatre stratégies d'altération des caractéristiques des sources de nourriture sont définies selon des schémas algorithmiques différents. Ces schémas sont explicités comme suit :

- **La stratégie d'altération "Single" :** adopte un mécanisme de mise à jour des solutions modifiant un seul paramètre choisi aléatoirement, ce qui correspond à un déplacement unidimensionnel en changeant une seule dimension choisie aléatoirement.
- **La stratégie d'altération "Full" :** qui consiste à modifier toutes les dimensions d'une source de nourriture simultanément.
- **La stratégie d'altération "Random" :** qui sélectionne aléatoirement le nombre de paramètres à faire évoluer simultanément dans l'intervalle $[1, D]$.
- **La stratégie d'altération "Combined" :** qui combine les mécanismes "Single", "Full" et "Random" selon le planning suivant : les abeilles Employed sont gérées pour commencer la recherche en changeant une seule dimension aléatoire. Ensuite, lorsque le nombre d'itérations

atteint $iter = MCN/5$, l'algorithme aura à sélectionner aléatoirement dans l'intervalle $[1, D]$ le nombre de paramètres à faire évoluer. À la fin du processus d'optimisation, entre $iter = \frac{4}{5}MCN$ et $iter = MCN$, les abeilles Employed mettront à jour toutes les dimensions de la solution en cours. Par la suite, après l'achèvement de la phase Employed, les trois sous-groupes d'abeilles Onlooker commenceront à mettre à jour une seule dimension de la solution en cours choisie aléatoirement. Puis, lorsque $iter = MCN/5$, les abeilles Onlooker sélectionneront aléatoirement le nombre de paramètres à optimiser dans l'intervalle [1, D].

Les résultats comparatifs sont illustrés dans le tableau III.15. Il en découle que l'association de la stratégie d'altération des caractéristiques des sources de nourriture "Full" au modèle mCLABC réduit sa performance; alors que les stratégies "Single" et "Random" conduisent à un résultat légèrement supérieur ou équivalent sur un seul problème. En revanche, la stratégie "Combined" améliore les résultats pour les fonctions F1 et F15, tandis que le modèle de base arrive à atteindre les minimas globaux des fonctions F11 et F13.

La stratégie d'altération des caractéristiques des sources de nourriture par les abeilles est déterminante pour le processus d'exploitation. La diversification des solutions est influencée par la stratégie adoptée, ce qui explique les résultats obtenus pour les différentes procédures décrites ci-dessus. Ainsi, Il est important de remarquer l'effet du « planning » calqué au processus de recherche à travers le caractère sélectif induit dans les schémas considérés, qui a toutefois bien contribué à l'amélioration de la qualité des solutions trouvées.

Tableau III.15 : Analyse de robustesse du modèle mCLABC vis-à-vis du procédé d'altération des caractéristiques de la nourriture.

Fonction	Stratégie d'altération des caractéristiques de la nourriture									
	Single		Full		Random		Combined		mCLABC	
	Moy.	Éc.-Type	Moy.	Éc.-Type	Moy.	Éc.-Type	Moy.	Éc.-Type	Moy.	Éc.-Type
F1	3.60e-57	1.23e-56	5.85e-18	2.01e-17	9.07e-74	4.52e-73	**4.93e-75**	**2.20e-74**	8.08e-58	1.14e-57
F10	2.77e-01	7.46e-01	42.29	27.71	14.26	20.02	4.79e-01	1.92	**5.88e-03**	**1.66e-02**
F11	**0**	**0**	32.07	8.86	7.66	2.54	2.37e-16	7.71e-16	**0**	**0**
F13	1.53e-09	8.35e-09	3.31e-01	4.76e-01	2.44e-02	2.66e-02	7.40e-18	4.05e-17	**0**	**0**
F15	3.30e-14	3.90e-15	7.49	1.68	1.20	8.97e-01	**1.08e-14**	**4.00e-15**	1.59e-14	3.19e-15

III.4.3 Impact du mécanisme de recherche

Pour examiner l'impact des règles de mise à jour des solutions sur la performance de la stratégie d'optimisation mCLABC proposée, nous avions défini trois différents mécanismes dénommés SS1 (Stratégie de recherche 1), SS2 (Stratégie de recherche 2), et SS3 (Stratégie de recherche 3). Le mécanisme SS1 utilise trois équations de recherche globale, à savoir

l'équation (III.7), l'équation (III.8), et l'équation (III.9) associées au sous-groupe d'abeilles Employed 1, 2 et 3, respectivement. Les sous-groupes 1, 2 et 3 d'abeilles Onlooker utilisent quant à elles les équations de recherche locale (III.10), (III.11), et (III.12), respectivement. Le mécanisme de recherche implique itérativement toutes ces équations de recherche lors du processus de recherche de solutions.

Pour le mécanisme de recherche SS2, les mêmes équations de recherche définies pour le mécanisme SS1 sont employées, sauf que la procédure d'appel d'une règle de mise à jour pour l'itération en cours est établie de sorte qu'une équation donnée puisse ne pas être impliquée ou bien impliquée plusieurs fois selon une probabilité donnée.

Enfin, le mécanisme SS3 utilise les équations (III.7), (III.10) et (III.11) pour les sous-groupes d'abeilles employed 1, 2 et 3, respectivement, et les équations (III.8), (III.9) et (III.12) pour les sous-groupes d'abeilles Onlooker 1, 2 et 3, respectivement.

La stratégie d'altération des sources de nourriture considérée pour les tests menés est celle décrite pour le modèle mCLABC. Les résultats comparatifs sont indiqués dans le tableau III.16. Il en ressort que les mécanismes de recherche SS1, SS2, SS3 préservent dans l'ensemble la performance de la stratégie d'optimisation proposée. En effet, le modèle mCLABC demeure performant sur la plupart des cas testés à l'exemple des fonctions F1, F10, F11 et F13 avec des minimas globaux parfaitement atteints dans les deux derniers cas.

Tableau III.16 : Analyse de robustesse du modèle mCLABC vis-à-vis du mécanisme de recherche à règles multiples.

Fonction	Mécanisme de recherche multiple							
	SS1		SS2		SS3		mCLABC	
	Moy.	Éc.-Type	Moy.	Éc.-Type	Moy.	Éc.-Type	Moy.	Éc.-Type
F1	4.91e-57	8.57e-57	1.11e-56	1.82e-56	5.21e-56	2.10e-55	**8.08e-58**	**1.14e-57**
F10	1.05e-01	5.38e-01	2.80e-01	1.48e+00	2.79e-01	1.04e+00	**5.88e-03**	**1.66e-02**
F11	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
F13	**0**	**0**	4.93e-04	1.88e-03	2.47e-04	1.35e-03	**0**	**0**
F15	1.58e-14	4.32e-15	1.44e-14	3.54e-15	**1.27e-14**	**3.77e-15**	1.59e-14	3.19e-15

III.5 Conclusion

Ce chapitre est dédié exclusivement à la nouvelle approche d'optimisation à base du modèle ABC que nous proposons dans le cadre de ce travail. Ainsi, deux stratégies améliorées dénommées CLABC et mCLABC sont développées et validées à travers une simulation intensive impliquant des analyses de performance et de robustesse.

La conception de l'approche proposée est basée sur une perception différente du modèle comportemental des essaims d'abeilles lors du fourragement, qui emploie de nouveaux concepts liés à l'apprentissage coopératif, la recherche à règles multiples et l'altération des caractéristiques des sources de nourriture, dont la modélisation aurait nécessité l'introduction de modifications structurelles et paramétriques à la version de base.

L'analyse de performance est menée à travers une étude en simulation étendue illustrant des résultats comparant les performances de la stratégie proposée au modèle ABC de base et ses principales variantes, ainsi qu'à d'autres métaheuristiques avancées. Un large panel de fonctions est considéré dans les différents tests, à l'exemple des fonctions numériques standards, et des fonctions complexes du type CEC'05 et CEC'13. L'analyse de performance de l'approche proposée est appuyée par une étude statistique commune à ce genre de problèmes. Les résultats obtenus affichent dans l'ensemble une nette supériorité des modèles d'optimisation développés, dont la performance revient en grande partie aux différents mécanismes introduits dans le processus de recherche et qui ont considérablement impacté les aspects de diversification et d'intensification qui interviennent à la fois dans les procédés d'exploitation et d'exploration. L'applicabilité de l'approche proposée à différents problèmes scientifiques fera l'objet du chapitre suivant.

Chapitre IV

Étude de cas d'applications

IV.1 Introduction

L'étude de performance et de robustesse de l'approche d'optimisation proposée présentée dans le Chapitre III sur des problèmes d'optimisation numériques complexes a permis de faire ressortir des conclusions majeures quant à la faisabilité de l'approche et son utilité en tant qu'outil d'optimisation prometteur. Les problèmes d'optimisation de fonctions numériques catégorisées dans des bases standards ou benchmarks représentent souvent des exemples de manipulation qui peuvent être calqués sur des problèmes réels du domaine de l'ingénierie des systèmes. Par la considération d'applications typiques, l'approche d'optimisation gagnerait plus de visibilité au sens des caractéristiques du problème réel appréhendé.

C'est dans cette optique que nous convenons d'examiner dans ce chapitre la validation de l'approche mCLABC proposée sur six applications relevant de différents domaines de l'ingénierie des systèmes. Il s'agit tout précisément des problèmes scientifiques suivants :

- Identification des systèmes neuro-flous
- Conception des classifieurs neuronaux
- Classification de données non supervisée
- Conception mécanique sous contraintes
- Conception d'un compensateur de non linéarité pour le transducteur LVDT
- Optimisation du contrôle PID

Comme il peut être constaté, certaines applications appréhendées relèvent du domaine des systèmes intelligents auxquels sont associés des problèmes d'optimisation difficile impliquant des bases de données de grande dimension.

IV.2 Identification des systèmes neuro-flous

L'identification des systèmes est fondamentalement un processus de génération de modèles comportementaux utilisés pour prédire les dynamiques à partir des signaux d'entrée et de sortie. Au cours de ces dernières années, les études sur l'utilisation des réseaux neuro-flous

adaptatifs (ANFIS) à des fins d'identification se sont vues progressées de manière considérable [94].

L'identification à base de réseaux neuro-flous adaptatifs (ANFIS) se fond sur l'utilisation de modèles combinant les caractéristiques du système d'inférence floue (FIS) et du réseau de neurones artificiel (ANN). La conception du modèle ANFIS peut être formulée comme un problème d'optimisation dont l'objectif est la détermination des paramètres optimaux de la prémisse et de la conséquence des règles floues. Ainsi défini, la résolution du problème implique un processus d'apprentissage faisant évoluer les paramètres à optimiser moyennant un algorithme d'optimisation. Deux principales approches sont à distinguer : la première approche emploie des algorithmes d'optimisation à descente de gradient. La recherche de solutions optimales à l'aide de méthodes à descente de gradient est souvent qualifiée de locale. La deuxième approche fait appel à des méthodes heuristiques de recherche globale à l'exemple des algorithmes génétiques (GA), les essaims particulaires (PSO), les systèmes immunitaires artificiels (AIS), les colonies de fourmis (ACO), les systèmes de fourragement bactériens (BFOA), l'algorithme d'abeilles (BA), la recherche coucou (CS), l'évolution différentielle (DE), la recherche harmonique (HS) et les colonies d'abeilles artificielles (ABC). Chacune de ces techniques a fait l'objet d'étude récente traitant le problème particulier d'apprentissage du modèle ANFIS [95].

Dans cette première application, nous nous sommes fixés l'objectif d'étudier la performance de la stratégie d'optimisation proposée sur le problème d'apprentissage du réseau ANFIS en considérant deux problèmes d'identification de systèmes dynamiques non linéaires distincts repris des travaux présentés dans [95].

IV.2.1 Architecture du réseau neuro-flou adaptatif (ANFIS)

Le modèle ANFIS «Adaptive Network Based Fuzzy Inference System» est un réseau *Neuro-Flou* adaptatif proposé initialement par Jang dans [96]. A titre d'illustration, l'architecture du réseau ANFIS, telle que représenté par la figure IV.1, comprend deux entrées x et y, une seule sortie z et cinq couches de neurones, désignées comme suit :

Couche 1: "Couche de fuzzification"

Les nœuds de cette première couche sont caractérisés par des fonctions d'appartenance qui peuvent prendre plusieurs formes (triangulaire, gaussienne, ...). Pour les problèmes étudiés, des fonctions d'appartenance de type cloche sont utilisées. Elles sont définies par :

$$\mu_{A_i}(x) = gbellmf(x; a, b, c) = \frac{1}{1+\left|\frac{x-c}{a}\right|^{2b}} \quad \text{(IV.1)}$$

$$O_i^1 = \mu_{A_i}(x) \quad \text{(IV.2)}$$

Les paramètres $\{a, b, c\}$ sont dits paramètres de la prémisse.

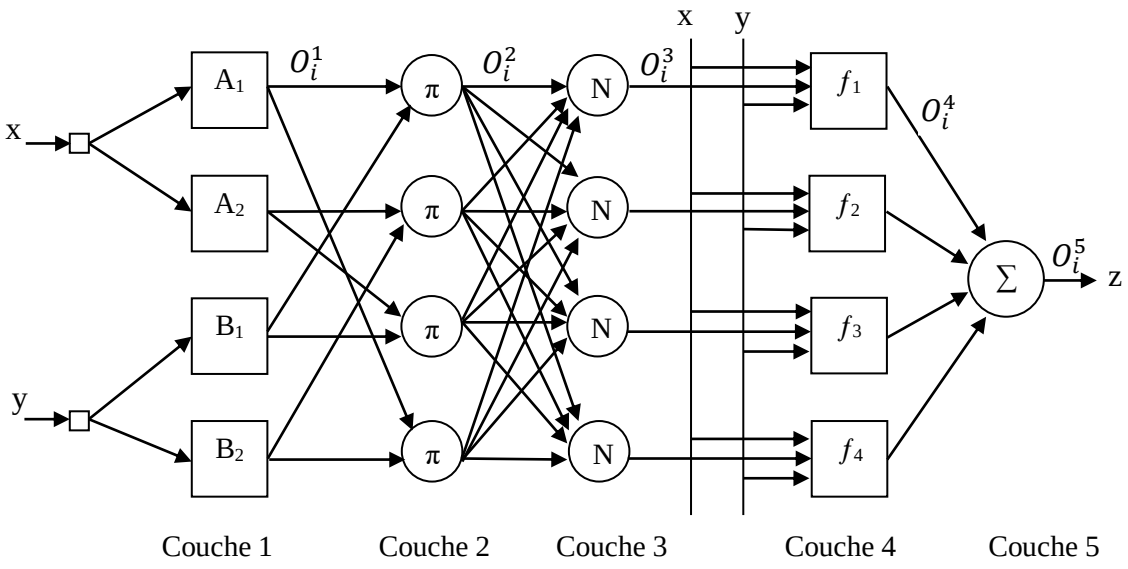


Figure IV.1 : Architecture du modèle ANFIS à deux entrées, une sortie et deux fonctions d'appartenance par entrée

Couche 2: "Couche des règles floues"

Cette couche sert à calculer le degré d'activation de chaque règle générée dans la couche de fuzzification. L'opérateur produit est généralement utilisé, ce qui donne en sortie :

$$O_i^2 = w_i = \mu_{A_i}(x) \cdot \mu_{B_i}(y), \qquad i = 1,2 \quad \text{(IV.3)}$$

Couche 3: "Couche de normalisation"

Au niveau de cette couche s'effectue la normalisation du degré d'activation w_i de chaque règle floue i, selon le formalisme suivant :

$$O_i^3 = \overline{w_i} = \frac{w_i}{w_1 + w_2} \qquad i = 1,2 \quad \text{(IV.4)}$$

Couche 4: "Couche de défuzzification"

Chaque nœud i de cette couche est un nœud adaptatif dont la sortie est :

$$O_i^4 = \overline{w_i} f_i = \overline{w_i}(p_i x + q_i y + r_i) \quad \text{(IV.5)}$$

où $\overline{w_i}$ est la sortie de la couche de normalisation, et $\{p_i,\ q_i,\ r_i\}$ est l'ensemble des paramètres. Ces paramètres sont appelés paramètres de la conséquence.

Couche 5: "Couche de sommation"

Cette couche est composée d'un seul nœud au niveau duquel s'effectue la somme des sorties provenant de la couche de défuzzification. Il en résulte alors :

$$O_i^5 = \sum_i \overline{w_i} f_i = \frac{\sum_i w_i f_i}{\sum_i w_i} \quad \text{(IV.6)}$$

IV.2.2 Apprentissage du réseau ANFIS par le modèle mCLABC

- *Problème d'identification 1 :* Le système dynamique non linéaire à identifier est décrit par le modèle discret :

$$y_p(k+1) = 0.72y_p(k) + 0.025y_p(k-1)u(k-1) + 0.01u^2(k-2) + 0.2u(k-3) \quad \text{(IV.7)}$$

avec

$$u(k) = \sin(2\pi k/25) \quad \text{(IV.8)}$$

où $u(k)$ est l'entrée du système et $y_p(k)$ sa sortie. Pour résoudre ce problème d'identification, nous utiliserons un modèle ANFIS à deux entrées $y(k)$ et $u(k)$ et une seule sortie $y(k+1)$, au même titre que le modèle considéré dans [95].

Chacune de ces entrées possède deux fonctions d'appartenance, formant ainsi quatre règles. Nous aurions ainsi au total 24 paramètres à optimiser (12 paramètres de la prémisse et 12 paramètres de la conséquence). L'ensemble des données considérées dans le test d'identification contient 1000 échantillons, dont 900 échantillons sont employés pour la phase d'apprentissage et les 100 données restantes sont utilisées pour la phase de validation.

- *Problème d'identification 2 :* Le système non linéaire est décrit par le modèle :

$$y_p(k+1) = \frac{y_p(k)y_p(k-1)[y_p(k)+2.5]}{1+y_p(k)^2+y_p(k-1)^2} + u(k) \quad \text{(IV.9)}$$

où $y_p(k)$ désigne la sortie du système et $u(k)$ est son entrée donnée par l'équation (IV.8).

Pour ce problème, la structure du réseau ANFIS comprend trois entrées $u(k)$, $y(k)$ et $y(k-1)$, et une seule sortie $y(k+1)$. Deux fonctions d'appartenance sont considérées pour chaque entrée, formant ainsi huit règles floues. Il en résulte 50 paramètres au total à optimiser (18 paramètres de la prémisse et 32 paramètres de la conséquence). L'ensemble des données considérées regroupe 250 échantillons, dont 200 échantillons sont employés pour la phase d'apprentissage et les 50 restants sont utilisés pour la validation.

L'objectif assigné au processus d'identification étant de construire pour chaque exemple étudié un modèle neuro-flou adaptatif ANFIS caractérisant au mieux la dynamique du système. L'application de la stratégie mCLABC aux problèmes considérés repose sur les mêmes considérations explicitées dans [95]. Les résultats sont comparés à ceux obtenus par l'emploi des approches ABC, aABC (Adaptive and hybrid ABC) [95] et mHPGA (Modified high performance GA) [50]. La fonction objectif à minimiser est donnée par la racine de l'erreur quadratique moyenne «Root Mean Squared Error (RMSE)» :

$$f_{obj} = RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2} \quad \text{(IV.10)}$$

où $\widehat{y_i}$ représente la sortie du modèle ANFIS et y_i la sortie du système, et N est le nombre de données disponibles (échantillons). Pour une comparaison fiable, nous avions considéré les mêmes paramètres de contrôle utilisés par les algorithmes comparés. La taille de la colonie est fixée à 20, la valeur *limit* est 250, et le nombre d'itérations maximal est 5000 pour le modèle ABC et 2000 pour les autres approches. Chaque test est déroulé 30 fois.

IV.2.4 Résultats d'application

Les résultats comparatifs sont indiqués dans le tableau IV.1. Ils sont exprimés en termes d'erreurs d'approximation moyennes (RMSE apprentissage) et de prédiction (RMSE validation). Comme il peut être clairement constaté, les résultats obtenus à l'aide de l'approche mCLABC proposée sont plus performants que ceux obtenus par l'utilisation des algorithmes ABC, aABC et mHPGA, tant pour les données d'apprentissage que pour les données de validation.

Les figures IV.2a et IV.2c comparent la sortie du processus avec celle du modèle identifié pour les deux problèmes étudiés. Les erreurs d'identification instantanées résultantes ($e = \hat{y} - y$) sont représentées en figures IV.2b et IV.2d.

Tableau IV.1 : Résultats d'application de l'approche mCLABC à l'apprentissage du réseau ANFIS

Algorithme	Problème 1			Problème 2		
	Nombre de paramètres	RMSE apprentissage	RMSE validation	Nombre de paramètres	RMSE apprentissage	RMSE validation
ABC	24	0.0244	0.0236	50	0.0224	0.0219
aABC	24	0.0232	0.0202	50	0.0152	0.0146
mHPGA	24	0.0074	0.0049	50	0.0125	0.0115
mCLABC	24	**0.0055**	**0.0023**	50	**0.0113**	**0.0108**

A l'issue de ces résultats, il convient de constater qu'une parfaite correspondance entre sortie réelle et sortie prédite par le réseau neuro-flou ANFIS est réalisée pour les deux problèmes d'identification. La qualité d'approximation obtenue valide ainsi la faisabilité et la performance de l'approche proposée.

IV.3 Conception des classifieurs neuronaux

La classification supervisée constitue un problème très répandu et largement étudié dans plusieurs domaines tels que la prédiction des pannes, l'analyse de données, la catégorisation de textes et la reconnaissance des formes. Elle représente les techniques qui apprennent à classer, à partir d'un échantillon de données déjà classées, de nouvelles données du même type que l'échantillon. Il existe de nombreuses techniques de classification supervisée, dont les

principales utilisent les réseaux de neurones. Grâce à leur capacité d'apprentissage, les réseaux de neurones permettent de construire des relations complexes non linéaires entre un grand nombre de variables, avec le seul recours aux situations apprises. De ce fait, ils sont largement impliqués dans la résolution de problèmes de classification [97]. Les réseaux de neurones non bouclés «feed-forward ANN» en représentent un exemple typique de classifieur neuronal [98]. La littérature en recense un grand nombre d'applications faisant appel aux réseaux de neurones multicouches «multilayer feed-forward ANN» pour la classification des données [99].

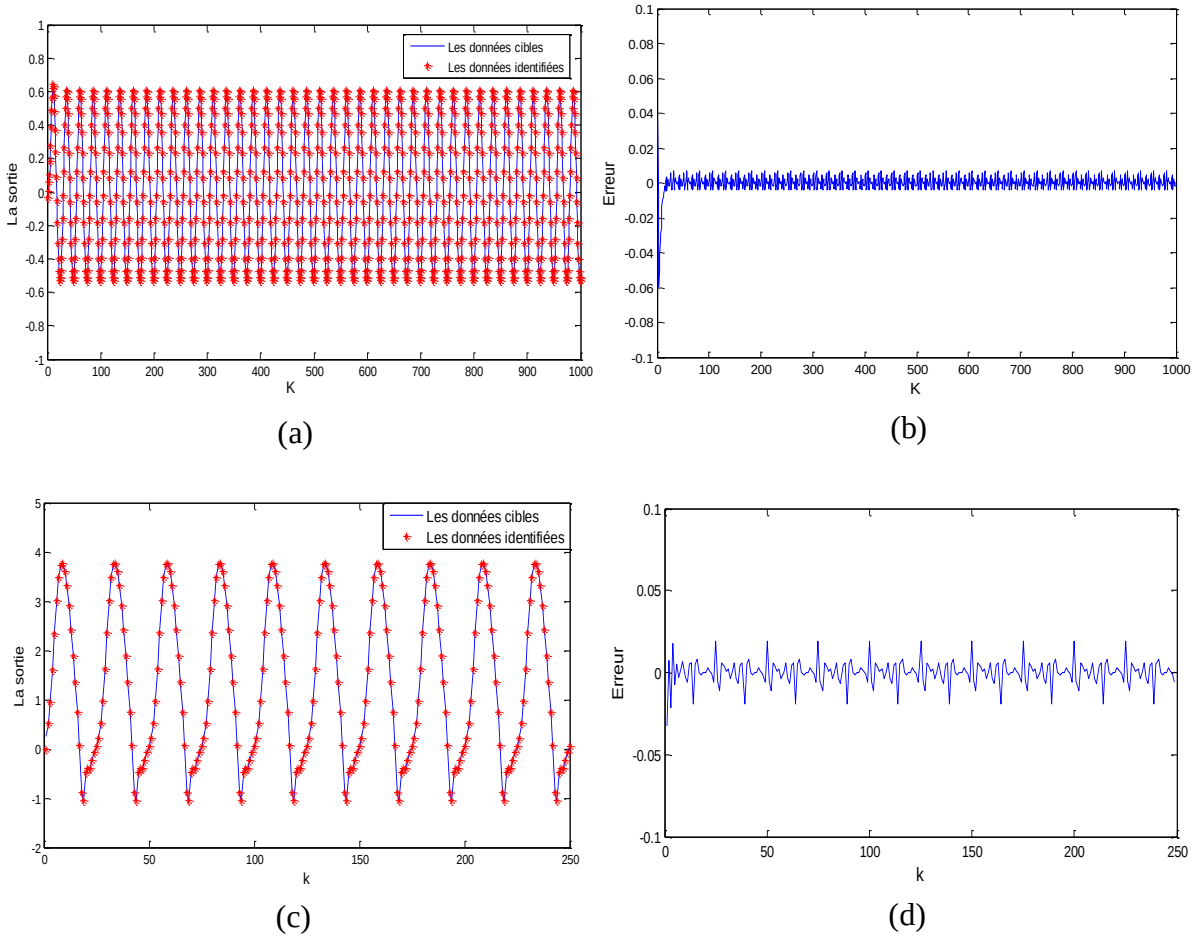


Figure IV.2 : Performance du modèle mCLABC sur les problèmes d'identification à base du réseau neuro-flou ANFIS : a) la sortie réelle et la sortie du modèle ANFIS pour le problème 1, b) l'erreur d'identification pour le problème 1, c) la sortie réelle et la sortie du modèle ANFIS pour le problème 2, d) l'erreur d'identification pour le problème 2.

Pour qu'un réseau de neurones non bouclé effectue une classification correcte, une configuration préalable est requise. En effet, il y a lieu d'abord de spécifier l'architecture du réseau classifieur, puis déterminer les poids et les biais de connexion. La conception de

l'architecture du réseau classifieur se fond généralement sur l'expérience appuyée souvent par des essais « trail and error » [100]. L'objectif de cette application est de déterminer la combinaison optimale des poids de connexion et des biais d'un réseau de neurones non bouclé afin de réaliser une tâche de classification.

Dans ce sens, deux modes de fonctionnement peuvent être distingués. Dans le premier mode, les paramètres du réseau sont ajustés à partir d'exemples pour lesquels on connaît la réponse désirée; c'est la «phase d'apprentissage». Dans le deuxième mode, le réseau est manipulé en présence de situations non apprises; c'est la «phase de généralisation» [101]. La dimension du problème est aussi déterminante, le temps de calcul et l'espace mémoire nécessaire en dépendent largement.

Le problème d'apprentissage des classifieurs neuronaux est perçu comme un problème d'optimisation difficile. Les algorithmes d'apprentissage déterministes (de recherche locale) sont employés pour le résoudre. Néanmoins, la performance résultante du classifieur est très limitée en raison de la qualité des solutions obtenues. Les métaheuristiques se sont alors envisagées comme alternative vu le caractère stochastique du processus de recherche. Parmi les techniques de recherche globale utilisées pour la classification des données, nous pouvons citer les essaims particulaires PSO [102], les algorithmes génétiques GA [103], l'évolution différentielle DE [104] et les colonies d'abeilles ABC. L'algorithme ABC s'est montré imposant en termes d'efficacité dans les travaux présentés dans [64] et [105], dédiés à la conception de classifieurs neuronaux.

Dans la suite, nous présentons les résultats d'application de la nouvelle approche mCLABC proposée au problème de conception de classifieurs neuronaux.

IV.3.1 Les réseaux de neurones artificiels

Les réseaux de neurones artificiels «Artificial Neural Networks (ANNs)» sont des modèles de calcul flexibles et non paramétriques dont la conception est inspirée des neurones biologiques. Un réseau de neurones artificiel est constitué d'un ensemble de neurones (nœuds) connectés entre eux par des liens qui permettent de faire propager les signaux d'un neurone à un autre dans une configuration de réseau. Chaque neurone calcule une sortie unique en se basant sur les informations qui lui sont transférées, comme le montre la figure IV.3.

Un réseau de neurones multicouche est composé d'une succession de couches dont chacune prend ses entrées sur les sorties de la précédente. La topologie n-p-m signifie l'existence de 3 couches : la couche d'entrée à n neurones, la couche cachée à p neurones et la couche de sortie à m neurones. La figure IV.4 illustre la structure d'un réseau de neurones à une seule couche

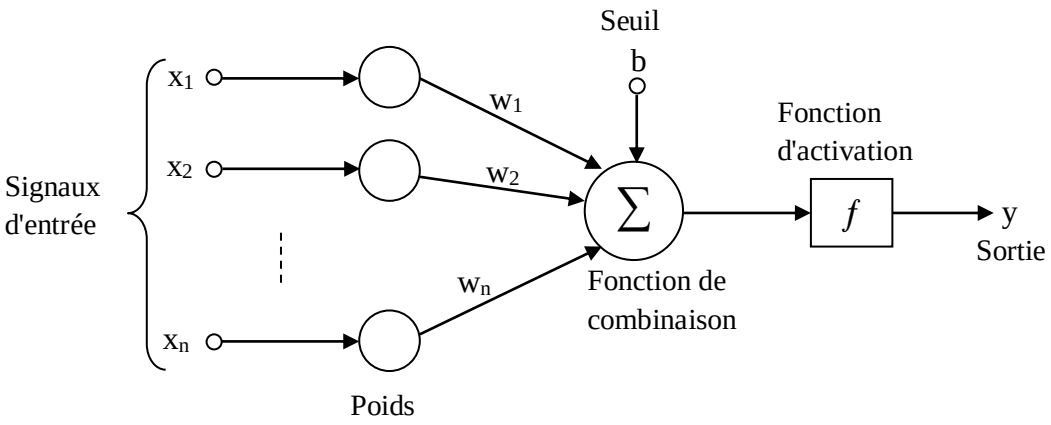


Figure IV.3 : Modèle d'un neurone artificiel.

cachée donné par la topologie 3-3-2. La sortie du i[ième] neurone est donnée par l'équation suivante:

$y_i = f_i\left(\sum_{j=1}^{n} w_{ij} x_j + b_i\right)$ $\quad i = 1,2,\ldots n$: nombre d'entrées (IV.11)

où y_i est la sortie du nœud, x_j représente la j[ième] entrée du nœud, w_{ij} est le poids synaptique associé au i[ème] neurone de la couche d'entrée et le j[ième] neurone de la couche de sortie, et b_i est le paramètre du nœud appelé aussi biais. La fonction f_i désigne la fonction d'activation du neurone. Elle peut prendre plusieurs formes dont la forme sigmoïdale s'exprime par :

$y = f(net) = \frac{1}{1+e^{-net}}$ (IV.12)

Un réseau de neurones nécessite une phase d'apprentissage par laquelle le réseau construit les liens qui lui permettent d'effectuer la classification des données présentées. Lors de cette phase, le réseau doit posséder un critère d'arrêt d'apprentissage.

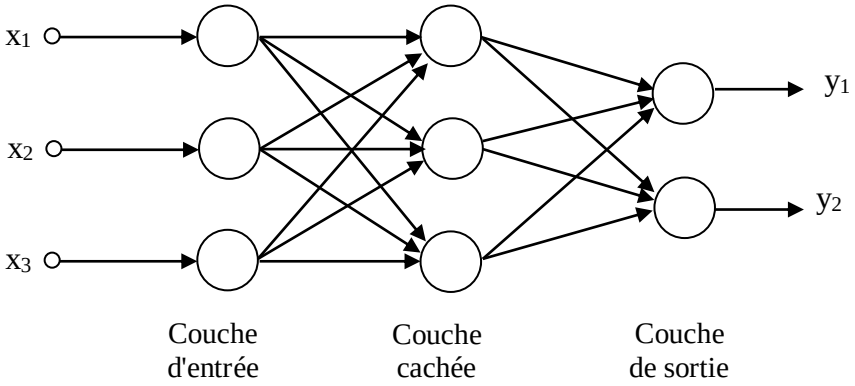


Figure IV.4 : Réseau de neurones multicouches à topologie 3-3-2.

IV.3.2 Apprentissage des classifieurs neuronaux par l'algorithme mCLABC

IV.3.2.1 Problèmes benchmarks de moyenne dimension

a) Description de la base de données

Les problèmes de classification de référence considérés dans ce test d'apprentissage sont : le problème OU-exclusif (XOR), le problème de parité de 3 bits (3-Bit Parity) et le problème d'encodeur-décodeur de 4 bits (Encoder/Decoder).

(1) Le problème OU- exclusif (XOR) : Il s'agit d'un problème non linéaire séparable dans lequel sont définies des fonctions booléennes mappant deux entrées binaires sur une seule sortie binaire. Les échantillons d'entrée-sortie de ce problème sont donnés par les combinaisons : (0 0; 0 1; 0 1; 1 1) → (0; 1; 1; 0).

Le classifieur neuronal à synthétiser pour ce type de problème est à topologie 2-2-1 caractérisé par six poids de connexion et trois biais, ce qui donne un total de 9 paramètres à optimiser.

(2) Le problème de parité de 3 bits (3bit parity): Le deuxième problème de classification à résoudre calcule le rèste de l'opérateur *ou exclusif* appliqué aux trois entrées binaires. En d'autres termes, si le nombre d'entrées binaires est impair, la sortie est égale à 1, sinon elle vaut 0. Les échantillons d'entrée-sortie définis pour le problème de parité de 3 bits sont donnés par les combinaisons:

(0 0 0; 0 0 1; 0 1 0; 0 1 1; 1 0 0; 1 0 1; 1 1 0; 1 1 1) → (0; 1; 1; 0; 1; 0; 0; 1).

Pour ce problème, le classifieur neuronal est à topologie 3-3-1, avec douze (12) poids de connexion et quatre (4) biais, soit seize (16) paramètres à optimiser.

(3) Le problème d'encodeur-décodeur de 4 bits (encoder/decoder) : Ce problème est représenté par quatre échantillons d'entrée distincts, chacun n'ayant qu'un bit activé. La sortie est une duplication des entrées. Les échantillons d'entrée-sortie de ce problème sont formulés par les combinaisons suivantes :

(0 0 0 1; 0 0 1 0; 0 1 0 0; 1 0 0 0) → (0 0 0 1; 0 0 1 0; 0 1 0 0; 1 0 0 0).

Cette configuration de données est assez proche des problèmes de classification de modèles du monde réel, où de petits changements dans les échantillons d'entrée entraînent de petits changements dans les échantillons de sortie [106]. Pour ce problème, on adopte une topologie de réseau de 4-2-4 avec dix-huit (18) poids de connexion et six biais, soit 22 paramètres au total.

b) Critère d'évaluation

L'apprentissage des modèles neuronaux définis pour ces problèmes est réalisé de manière supervisée, c'est-à-dire pour une entrée particulière présentée au réseau correspondrait une sortie désirée. L'ajustement des poids s'effectue en comparant la sortie résultante du réseau avec la sortie désirée. La fonction d'évaluation pour le modèle de classification est définie comme suit :

$$E\big(w(t), b(t)\big) = \frac{1}{N}\sum_{i=1}^{m}\sum_{j=1}^{N}(y_i - o_i)^2 \qquad \text{(IV.13)}$$

où $E\big(w(t), b(t)\big)$ représente l'erreur quadratique moyenne (MSE) à l'itération t, $w(t)$ et $b(t)$ sont les poids et les seuils associés aux connexions à l'itération t, y_i et o_i représentent la sortie du réseau et la sortie désirée du $ième$ nœud de sortie, m est le nombre de nœuds de sortie et N est le nombre de données.

c) Résultats de classification

L'apprentissage des modèles neuronaux définis pour les problèmes de classification de moyenne dimension est effectué séparément moyennant l'algorithme de rétro-propagation à descente du gradient (GD-BP), l'algorithme de rétro-propagation basé sur la loi de Lenvenberg-Marquant (LM-BP), l'algorithme Bat (BA), l'algorithme de recherche Cuckoo (CS), l'algorithme ABC, l'algorithme Bat modifié (MBA) et l'algorithme mCLABC. Un ensemble de 30 essais est considéré pour chaque problème avec un critère d'arrêt défini sur l'erreur quadratique moyenne (MSE ≤ 0,01) ou sur le nombre maximal de cycles (MCN). Pour une comparaison fiable, les méthodes comparées sont utilisées avec les mêmes paramètres de contrôle communs utilisés dans [24]. Pour les modèles ABC et mCLABC, la taille de la colonie est fixée à 50 et le paramètre "$limit$" est égal à $SN * D$. L'intervalle de variation des paramètres (gamme), la dimension (D), le nombre maximal de cycles (MCN) et les topologies des réseaux sont indiqués dans le tableau IV.2.

Tableau IV.2 : Paramètres des problèmes de classification benchmarks de moyenne dimension.

Problème	Gamme	Topologie	D	MCN
XOR	[-10, 10]	2-2-1 + Biais (3)	9	100
3-Bit Parity	[-10, 10]	3-3-1 + Biais (4)	16	1000
Enc. /Dec.	[-10, 10]	4-2-4 + Biais (6)	22	1000

Les résultats comparatifs des algorithmes GD-BP, LM-BP, BA, CS, ABC, MBA et mCLABC appliqués aux problèmes de classification XOR, 3 bits de parité et 4 bits d'encodeur-décodeur sont représentés dans le tableau IV.3. Ils sont exprimés en termes de

moyenne des erreurs quadratiques moyennes (MMSE), d'écart type des erreurs quadratiques moyennes (SDMSE), du nombre moyen d'époques (ME) pour les algorithmes de rétro-propagation, du nombre moyen de cycles (MC) pour les algorithmes évolutionnaires et du taux de réussite (SR). Les meilleurs résultats sont marqués en gras.

Tableau IV.3 : Résultats d'application de l'approche mCLABC à la classification neuronale des problèmes XOR, 3Bit parity et encoder/decoder.

Algorithme		XOR	3-Bit Parity	Enc./Dec.
GD-BP	MMSE	0.212	0.2493	0.0809
	SDMSE	0.0369	0.0025	0.0756
	ME	500	1600	2020
	SR	0	0	0
LM-BP	MMSE	0.0491	0.0209	0.0243
	SDMSE	0.0646	0.043	0.0424
	ME	13	21.333	83.3667
	SR	66.66	86.66	73.33
BA	MMSE	0.006275	0.006924	0.007601
	SDMSE	0.003448	0.002194	0.001825
	MC	60	325.1	300.1667
	SR	86.6667	100	100
CS	MMSE	0.005913	0.006565	0.007285
	SDMSE	0.002351	0.002120	0.001808
	MC	30.96667	212.1	219.86667
	SR	100	100	100
ABC	MMSE	0.006956	0.006679	0.008191
	SDMSE	0.002402	0.002820	0.001864
	MC	32	179.07	185
	SR	100	100	100
MBA	MMSE	0.004656	0.005432	0.006864
	SDMSE	0.002873	0.002986	0.002338
	MC	**26.233**	**137.4**	184.37
	SR	100	100	100
mCLABC	MMSE	**5.59931e-05**	**4.91111e-05**	**1.53255e-04**
	SDMSE	**7.37792e-06**	**4.99936e-05**	**9.44449e-10**
	MC	33	174	**110**
	SR	**100**	**100**	**100**

D'après les résultats obtenus, nous pouvons constater que l'approche proposée arrive à améliorer nettement les valeurs moyennes des erreurs quadratiques moyennes (MMSE) et d'écart-types des erreurs quadratiques moyennes (SDMSE) pour l'ensemble des problèmes de classification étudiés. Ces résultats sont atteints sur des cycles moyens différents avec un taux de réussite de 100%.

IV.3.2.2 Problèmes benchmarks de grande dimension

a) Description de la base de données

Les bases de données de l'UCI (University of California, Irvine) [107] regroupe une collection de données de grande dimension utilisée largement dans les études de classification. Pour ce cas d'application, nous considérons trois bases désignées par les attributs « Cancer », « Diabète » et « Cœur ». Les données de chacune de ces classes sont catégorisées en deux sous-ensembles : le premier regroupe 75% des données utilisées pour l'apprentissage des classifieurs neuronaux, le deuxième en contient 25% des données restants, et qui sont utilisées dans la phase de généralisation.

(1) La base cancer «Cancer» **:** correspond aux données du diagnostic du cancer de seins (une tumeur est classée comme bénigne ou maligne). L'ensemble de données contient 699 instances (observations), chacune comprenant 9 entrées et 2 sorties. Les premières 525 instances sont utilisées pour l'apprentissage du réseau et les 174 instances restantes sont utilisées pour le test.

(2) La base diabète «Diabetes» : correspond aux données du diagnostic de diabète chez les personnes. L'ensemble de données contient 768 instances, chacune comprenant 8 entrées et 2 sorties. Pour cette classe, 576 instances sont marquées comme données d'apprentissage et les 192 restantes comme données de test.

(3) La base cœur «Heart» : correspond aux données du diagnostic d'une maladie cardiaque (établi sur la base de l'observation de la réduction de plus de 50% des diamètres des quatre principaux vaisseaux cardiaques). L'ensemble de données contient 920 instances, chaque instance ayant 35 entrées et 2 sorties. Les 690 premières instances forment les données d'apprentissage et les 230 restantes représentent les données de test.

b) Critère d'évaluation

Pour ce type de problèmes, l'adaptation des poids du classifieur neuronal est réalisée en minimisant (optimisant) la fonction d'erreur du réseau, appelée pourcentage de l'erreur carrée (SEP), qui s'exprime par :

$$SEP = 100\frac{O_{\max} - O_{\min}}{P.N}\sum_{j=1}^{P}\sum_{k=1}^{N}\left(d_k - o_k\right)^2 \qquad \text{(IV.14)}$$

où $O_{\max}$ et $O_{\min}$ sont respectivement les valeurs maximale et minimale des signaux des nœuds de sortie, P représente le nombre d'échantillons, N est le nombre de classes, d_k et o_k représentent respectivement les valeurs désirée et réelle du nœud de sortie.

c) Résultats de classification

Dans cette étude de performance, la nouvelle approche proposée est comparée à des méthodes déterministes, en l'occurrence l'algorithme de rétro-propagation [108] et l'algorithme Levenberg-Marquardt [109], et des méthodes de recherche globale à l'instar de l'algorithme ABC, l'algorithme GA [12], l'algorithme DE [110], l'algorithme PSO [111], l'algorithme ABC optimisé Opt.ABC [112] et l'algorithme ABC optimisé sans contraintes Opt.UABC [112].

Les classifieurs à concevoir sont représentés par des modèles neuronaux non bouclés à une seule couche cachée. Le nombre de neurones dans les couches d'entrée et de sortie dépend des modèles correspondants, tels que mentionné dans les caractéristiques des problèmes. Selon [105], il est recommandé d'utiliser une couche cachée à six neurones dans ce genre de problèmes de classification. Les fonctions d'activation sont de forme sigmoïdale avec des paramètres de nœuds (biasis) appliqués aux structures. Le tableau IV.4 résume les paramètres des problèmes de classification considérés.

Tableau IV.4 : Paramètres des problèmes benchmarks de grande dimension (real-world data)

Base de données	Structure	Dimension	Echantillons
Cancer	9-6-2	74	699
Diabetes	8-6-2	68	768
Heart	35-6-2	230	920

Les essais sont répétés 50 fois pour chaque problème appréhendé, le critère d'arrêt est défini sur le pourcentage d'erreur carrée (SEP $\leq$ 0,01) ou sur le nombre maximal de cycles (MCN). Le nombre maximal de cycles est fixé à 2000 et la limite d'espace de recherche est $[-2.0, 2.0]$. Le paramétrage des algorithmes comparés et les résultats qui leur sont associés sont repris des travaux publiés dans [105] et [112]. Ces mêmes paramètres sont utilisés par l'algorithme mCLABC, avec une taille de colonie fixée à 30 et une valeur limite égale à 1000.

Pour l'évaluation de performance, nous utilisons la moyenne et l'écart-type du pourcentage d'erreur de classification (CEP), qui correspond au pourcentage de modèles mal classés obtenus à partir de l'ensemble des données de test. Les modèles sont classés en affectant une classe à un neurone de sortie à l'aide de la méthode gagnant-prend-tout (Winner-takes-all), qui définie la classe par rapport à la sortie de la plus grande valeur. La sortie obtenue est ainsi comparée à la sortie désirée; l'instance est considérée mal classée en cas de sorties non identiques. Le pourcentage d'erreur de classification est calculé à travers l'équation suivante pour chaque ensemble de données de test :

$$CEP = 100 \times \frac{nombre\ d'instances\ mal\ classées}{nombre\ total\ d'instances} \quad \text{(IV.15)}$$

Les résultats exprimés en termes de moyenne (Moy.) et d'écart type (Éc.-Type) du pourcentage d'erreur de classification (CEP), sont illustrés dans le tableau IV.5. A première vue, l'approche proposée démontre son efficacité à travers la réduction remarquable du pourcentage d'erreur de classification (valeurs marqués en gras).

Tableau IV.5 : Résultats d'application de l'approche mCLABC à la classification neuronale des données Cancer, Diabetes et Heart

		BP	LM	ABC	PSO	DE	GA	Opt.ABC	Opt.UABC	mCLABC
Cancer	Moy.	1.89	8.18	**1.14**	2.01	1.19	1.35	-	-	**1.14**
	Éc.-Type	0.43	10.29	**0.00**	0.86	0.28	0.28	-	-	**0.00**
Diabetes	Moy.	28.27	29.80	25.22	27.50	24.84	26.51	-	-	**23.92**
	Éc.-Type	6.30	3.77	0.97	2.05	1.32	1.21	-	-	**0.74**
Heart	Moy.	21.44	24.96	19.48	21.87	20.33	20.60	19.13	19.49	**19.09**
	Éc.-Type	**0.55**	7.40	1.41	1.45	1.29	0.68	1.34	0.57	1.53

D'après les résultats obtenus, il en découle que le problème Cancer est le moins contraignant avec des résultats de classification raisonnablement bons pour la plupart des méthodes comparées. Plus précisément, l'algorithme ABC et l'algorithme mCLABC sont les plus performants avec une valeur moyenne de classification CEP de l'ordre de 1,14%, suivis de l'algorithme DE avec une moyenne de 1,19%, l'algorithme GA avec une moyenne de 1,35%, et les algorithmes BP et PSO avec des erreurs évaluées à 1,89%, et 2,01%, respectivement. L'algorithme LM affiche quant à lui une erreur de classification moyenne de 8,18%.

Le problème Diabète est en revanche un problème de classification plus difficile que le problème Cancer. Les entrées et le nombre de paramètres à optimiser sont cependant les mêmes pour les deux problèmes. Comme il peut être constaté, les performances globales réalisées ne sont pas aussi bonnes que celles obtenues pour le problème Cancer. Cependant, l'approche proposée arrive à surpasser tous les autres algorithmes comparés avec une erreur moyenne de classification CEP de l'ordre de 23.92%.

Enfin, pour le problème du diagnostic cardiaque de grande dimension, l'algorithme mCLABC achève une bonne performance dans le test de classification des données Heart avec une erreur moyenne CEP évaluée à 19,09%, qui est la meilleure erreur parmi toutes les autres valeurs enregistrées.

En somme, il convient de constater que les problèmes de classification neuronale de moyenne ou de grande dimension se trouvent mieux adaptés à la nouvelle approche

d'optimisation proposée, dont les performances se sont avérées nettement meilleures en réalisant des erreurs de classification minimales par rapport aux différentes méthodes déterministes et métaheuristiques comparées.

IV.4 Classification de données non supervisée

La classification de données non supervisée s'applique à une multitude de domaines dont l'exploration de données, l'analyse statistique, la compression de données, la segmentation d'images, l'identification et le contrôle des systèmes et bien d'autres. Appelée aussi « clustering », elle repose sur le concept de regroupement de données multidimensionnelles au sens de critères de similitude prédéfinis de sorte à former des classes [113].

Les méthodes de clustering se répartissent en deux catégories principales : les méthodes hiérarchiques et les méthodes basées sur le partitionnement des données. Les méthodes hiérarchiques se fondent sur l'idée de former des groupements de données suivant une procédure d'agrégation opérée sur les groupes de données les plus proches. Les méthodes de partition se basent généralement sur des critères de similitude exprimés en termes de distance, dont la minimisation produirait les classes recherchées. La méthode des K-moyennes « K-means » en est un exemple typique.

Le problème de clsutering de données est formulé en tant que problème d'optimisation et est résolu moyennant plusieurs approches, tant classiques de recherche locale [114] ou métaheuristiques de recherche globale [49] [113-117]. Dans ce même sens, l'étude sera portée sur l'utilisation de la nouvelle approche proposée pour la résolution du problème de classification de données non supervisée.

IV.4.1 Formulation du problème

La plupart des algorithmes de classification non supervisée utilisent une mesure de proximité entre les données à traiter. Cette proximité est formalisée à l'aide d'une mesure de similarité ou de dissimilarité entre les données. La mesure de distance est généralement utilisée pour évaluer la proximité entre les objets (données) [113].

Le clustering par une méthode évolutionnaire est généralement classé parmi les méthodes de partition. Dans une classification par méthode de partition, chaque objet appartient à une et une seule classe. L'ensemble de données Z est donc divisé en un ensemble de classes $C = \{c_1, c_2, \ldots, c_K\}$ formant une partition de Z.

On suppose que l'on dispose d'un ensemble d'objets que l'on dénote par $Z = \{z_1, z_2, \ldots, z_N\}$ caractérisé par un ensemble de descripteurs t, tel que $z_i = \{z_{i1}, z_{i2}, \ldots, z_{it}\}$. L'objectif du clustering consiste à trouver les classes auxquelles appartiennent chaque objet z_i que l'on désigne par $C = \{c_1, c_2, \ldots, c_K\}$. L'algorithme de classification procède par la recherche des centres des clusters afin de répartir les données en fonction de la distance évaluée par rapport au centre. La fonction objectif à minimiser est définie par l'équation (IV.16) :

$$f_i = \sum_{i=1}^{N} \sum_{j=1}^{K} min\left(Des\left(z_i, c_j\right)\right) \quad \text{(IV.16)}$$

où c_j représente le centre de la classe j, z_i est le i[ème] objet de données, N est le nombre total des objets, K est le nombre total de classes, et Des est la mesure de distance entre le vecteur d'objet z_i et le vecteur des centres de clusters c_j. Différentes mesures de distance peuvent être employées, à l'exemple de la distance euclidienne, qui est définie par l'équation (IV.17).

$$Des\left(z_1, c_2\right) = \sqrt{\sum_{j=1}^{t} \left(z_{1j} - c_{2j}\right)^2} \quad \text{(IV.17)}$$

où $Des\left(z_1, c_2\right)$ est la mesure de distance entre le premier objet et le deuxième cluster, z_{1j} est l'élément j du premier objet, et c_{2j} l'élément j du centroïde du deuxième cluster.

IV.4.2 Clustering à base du modèle mCLABC

a) Description de la base de données

Au total, cinq bases de données réelles issues de la base de référence de l'UCI [107] sont utilisées dans cette étude de performance. Les données considérées sont associées aux problèmes suivants :

(1) Iris : est une base de données sur la fleur Iris, qui contient 150 objets de données, 3 classes et 4 attributs.

(2) Seeds : est une base de données comprenant des grains appartenant à trois variétés différentes de blé, qui contient 210 objets de données, 3 classes et 7 attributs.

(3) Cancer : est une base de données qui contient 683 cas cliniques relatifs au cancer du sein répartis en 2 groupes avec 9 attributs. L'objectif étant de classer chaque cas en tumeurs bénignes ou malignes.

(4) Glass : est une base de données sur les types de verre, qui contient 214 objets de données de 6 différents types de verre, décrite par 9 attributs. L'objectif étant de classer ces données en 6 groupes.

(5) Wine : est une base de données qui contient 178 sortes de vins, décrite par 13 attributs. L'objectif est de les classer en 3 groupes.

Le tableau IV.6 résume les paramètres des problèmes de classification considérés.

b) Résultats de classification

L'étude comparative est menée sur les résultats de clustering obtenus par l'algorithme K-mean [118], l'algorithme PSO [117], l'algorithme PSO hybride (H-PSO) [49], l'algorithme de recherche d'harmonie (HS) [116]et l'algorithme de troupeau de krill hybride (H-KHA) [49]. Pour une comparaison fiable, les méthodes comparées sont utilisées avec les mêmes paramètres de contrôle communs, au même titre que ceux reportés dans [49]. Un ensemble de 20 essais est considéré pour chaque problème, le nombre maximal de cycles (MCN) est fixé à 1000 ($Max_FEs = 20000$). Pour le modèle mCLABC, la taille de la colonie est fixée à 20 et le paramètre *"limit"* est choisi égal à 100. La dimension de chaque problème est représentée dans le tableau IV.6.

Tableau IV.6 : Paramètres des problèmes de classification.

Base de données	Nombre d'attributs	Nombre de classes	Nombre d'instances	Dimension
Iris	4	3	150 (50, 50, 50)	12
Seeds	7	3	210 (70, 70, 70)	21
Cancer	9	2	683 (444, 239)	18
Glass	9	6	214 (70, 76, 17, 13, 9, 29)	54
Wine	13	3	178 (59, 71, 47)	39

Les résultats comparatifs du tableau IV.7 sont exprimés en termes de meilleures, moyennes et mauvaises valeurs de la fonction objectif à minimiser.

Le meilleur, le moyen et le mauvais pourcentage d'erreur de classification (EP), qui correspond au pourcentage des modèles mal classés obtenus à partir de l'ensemble de données de test pour chaque problème est aussi considéré lors de l'évaluation de performance. Il est donné par :

$$EP = \frac{nombre\ d'objects\ mal\ classés}{nombre\ total\ d'objets} \times 100 \quad \text{(IV.18)}$$

Les résultats obtenus sont illustrés dans le tableau IV.8.

Tableau IV.7 : Résultats comparatifs de classification de données non supervisée (clustering).

		H-KHA	H-PSO	K-mean	PSO	HS	mCLABC
Iris	Meilleure	96.154	96.590	97.325	**96.148**	98.648	96.655
	Moyenne	**96.524**	97.235	104.576	97.997	98.447	96.656
	Mauvaise	96.989	97.985	123.969	98.287	99.144	**96.657**
Seeds	Meilleure	4350.140	4395.445	4480.162	4532.142	4412.525	**311.798**
	Moyenne	4391.546	4390.900	4488.542	4607.564	4450.848	**311.810**
	Mauvaise	4399.254	4419.410	4499.649	4660.216	4460.464	**312.034**
Cancer	Meilleure	2975.191	2988.654	2989.258	3002.450	2988.856	**2964.387**
	Moyenne	2982.437	2990.263	2992.640	3012.394	2990.654	**2964.390**
	Mauvaise	2990.493	2993.372	3001.193	3090.910	2998.286	**2964.406**
Glass	Meilleure	213.105	213.162	216.198	279.767	243.157	**210.431**
	Moyenne	215.665	216.658	220.214	282.734	246.254	**212.147**
	Mauvaise	217.355	218.851	222.015	285.299	251.556	**215.201**
Wine	Meilleure	16350.154	16980.297	16642.141	16764.901	16759.449	**16292.185**
	Moyenne	16410.147	17587.145	16976.691	17964.390	16945.698	**16293.100**
	Mauvaise	16961.147	18564.154	20452.147	19297.145	16989.931	**16294.171**

Tableau IV.8 : Résultats comparatifs des erreurs de classification (EP).

		H-KHA	H-PSO	K-mean	PSO	HS	mCLABC
Iris	Meilleure	**9.000**	9.666	10.660	10.667	10.509	10.000
	Moyenne	19.866	15.800	21.467	15.867	21.054	**10.067**
	Mauvaise	43.333	44.333	56.667	43.447	44.286	**10.667**
Seeds	Meilleure	8.623	9.047	12.643	**8.571**	12.290	10.476
	Moyenne	11.666	13.881	12.643	15.262	13.015	**10.476**
	Mauvaise	14.523	19.238	12.643	36.190	16.021	**10.476**
Cancer	Meilleure	38.670	39.775	39.865	40.775	40.111	**3.514**
	Moyenne	39.012	39.125	42.388	43.051	42.054	**3.514**
	Mauvaise	44.154	46.758	45.970	45.455	45.640	**3.514**
Glass	Meilleure	**32.242**	41.589	42.262	43.925	41.162	45.327
	Moyenne	42.219	47.617	46.154	46.262	**42.054**	46.565
	Mauvaise	51.420	56.075	**46.215**	52.804	46.255	57.009
Wine	Meilleure	29.650	29.775	29.775	29.775	29.865	**28.090**
	Moyenne	33.000	30.871	32.388	32.051	32.568	**28.708**
	Mauvaise	42.134	43.888	43.820	44.449	44.467	**29.213**

D’après les résultats comparatifs montrés dans les tableaux IV.7-IV.8, il apparaît clairement que la stratégie proposée réalise visiblement les meilleurs performances sur la majorité des problèmes de clustering de données multidimensionnelles considérés, ce qui permet d’avancer une conclusion prometteuse quant à la faisabilité de l’approche en tant qu’outils d’optimisation, et à sa performance prouvée sur ces problèmes complexes de classification de données de grande dimension.

IV.5 Conception mécanique sous contraintes

En ingénierie des systèmes mécaniques, les problèmes de conception et de dimensionnement sont souvent formulés en tant que problèmes d'optimisation avec contraintes. Les méthodes appliquées pour les résoudre peuvent être de nature déterministes ou encore stochastiques. Les approches déterministes ont généralement besoin d'hypothèses de continuité et de différentiabilité des fonctions objectifs, ce qui est rarement satisfait en pratique. Les techniques d'optimisation stochastiques, quant à elles, se trouvent mieux adaptées aux problèmes avec contraintes car aucune hypothèse de continuité ou de différentaibilité de la fonction objectif n'a besoin d'être posée [119] [120] [61].

La résolution des problèmes d'optimisation contraints nécessite l'emploi de mécanismes appropriés de traitement des contraintes servant à orienter le processus vers des régions réalisables de l'espace de recherche. L'introduction des fonctions de pénalité constitue un des mécanismes utilisés dans ce sens; l'efficacité d'une telle approche est néanmoins tributaire d'un ajustement judicieux des facteurs de pénalité [121] [65]. L'approche de Deb [119] est un autre mécanisme de manipulation des contraintes qui a fait ses preuves avec le modèle ABC dans la résolution de problèmes d'optimisation avec contraintes [61] [122-124] et [65].

Nous proposons dans la suite d'examiner l'application de la nouvelle approche mCLABC à des problèmes de conception mécanique typiques formulés avec contraintes. Pour ce faire, quatre problèmes de conception sont considérés, dont deux problèmes à variables continues, un problème à variables discrètes et un dernier problème à variables mixtes.

IV.5.1 Adaptation de la structure algorithmique de l'algorithme mCLABC aux problèmes contraints

Pour la prise en compte des contraintes, le mécanisme de Deb est adopté. Etant donné qu'il soit parfois impossible de produire une solution réalisable de manière aléatoire [65], l'algorithme mCLABC, tel qu'a été initialement conçu dans le Chapitre III, ne considère pas une population initiale réalisable. L'adaptation de la structure algorithmique du modèle mCLABC est introduite de manière à ce que l'on arrive à diriger les solutions vers des régions réalisables au moyen de trois règles heuristiques de Deb [119], utilisées à la place de la *sélection gourmande*. La méthode de Deb utilise un opérateur de sélection de tournoi, où deux solutions sont comparées en fonction des critères suivants :

- Entre solution réalisable et solution non réalisable, la première est préférée,
- Parmi deux solutions réalisables, celle qui pénalise au mieux la fonction objectif est préférée,

- Parmi deux solutions non réalisables, celle qui présente le moins de violation de contraintes est préférée.

Le processus de production des abeilles Scout de l'algorithme mCLABC se base sur un mécanisme de diversification qui pourrait probablement insérer dans la population de nouvelles sources de qualité médiocre. Pour palier à cette situation, le modèle mCLABC adapté aux problèmes d'optimisation avec contraintes propose de produire des abeilles Scout artificielles à une période de cycles prédéterminée (SPP) en vue de générer aléatoirement de nouvelles sources de nourriture. À chaque cycle de SPP, il sera procédé au contrôle des sources de nourriture; une solution abandonnée déclencherait alors la phase Scout.

Outre ces adaptations, le calcul de probabilité est différent de celui adopté dans le modèle mCLABC dédié aux problèmes d'optimisation sans contraintes. Comme l'algorithme mCLABC n'a besoin de générer que des solutions réalisables lors de l'initialisation, et que des solutions non réalisables peuvent être ajoutées à la colonie via la phase Scout, une valeur de probabilité sera attribuée à chaque solution non réalisable de la colonie au même titre qu'une solution réalisable. Ainsi, l'algorithme mCLABC aura à produire des solutions réalisables en augmentant leurs probabilités, qui sont calculées en fonction de la proportion de leurs fitness par rapport à la somme des fitness associées à toutes les solutions réalisables. Dans la phase Onlooker, la probabilité de choisir un site de sources de nourriture est calculée pour une solution non réalisable en fonction de la proportion de la violation de contraintes évaluée par rapport à la somme de toutes les violations de contraintes. Les expressions suivantes définissent les probabilités pour chaque type de solution :

$$\mathrm{p_i} = \begin{cases} 0.5 + \left(\frac{\mathrm{fitness_i}}{\sum_{i=1}^{SN} \mathrm{fitness_i}}\right) * 0.5 & \text{si la solution est réalisable} \\ \left(\frac{\mathrm{CV}}{\sum_{i=1}^{SN} \mathrm{CV}}\right) * 0.5 & \text{si la solution est non réalisable} \end{cases} \quad \text{(IV.19)}$$

avec

$$\mathrm{CV} = \sum_{j=1,\ \text{if } g_j(x)>0}^{q} g_j(x) + \sum_{j=q+1}^{m} h_j(x) \quad \text{(IV.20)}$$

où $g_j(x)$ désigne la $j^{ième}$ contrainte d'inégalité, $h_j(x)$ est la $j^{ième}$ contrainte d'égalité, q est le nombre total des contraintes d'inégalité, et m le nombre total des contraintes.

IV.5.2 Application aux problèmes de conception mécanique

Quatre problèmes de conception mécanique très fréquents dans le domaine de l'optimisation avec contraintes seront étudiés. Il s'agit des cas d'étude suivants :

a) Problème du faisceau soudé : «welded beam»

Le problème du faisceau soudé illustré en figure IV.5a a pour objet de minimiser le coût de la poutre sous les contraintes de cisaillement (τ), de flexion de la poutre (σ), de la charge de flambage sur la barre (P_c), de déflexion de l'extrémité de la poutre (δ), et des contraintes latérales. Il existe quatre paramètres de conception x1, x2, x3 et x4 correspondant aux variables h, l, t et b, dont les désignations sont montrées en Figure IV.5a. Ainsi formulé, ce problème emploie une fonction objectif non linéaire, cinq contraintes inégalités non linéaires et deux contraintes inégalités linéaires. Il s'énonce comme suit :

$$Min\ f(x) = 1.10471 x_1^2 x_2 + 0.04811 x_3 x_4 (14.0 + x_2) \qquad \text{(IV.21)}$$

Tel que :

$$g_1(x): \tau(x) - \tau_{\max} \le 0$$

$$g_2(x): \sigma(x) - \sigma_{\max} \le 0$$

$$g_3(x): x_1 - x_4 \le 0$$

$$g_4(x): 0.10471 x_1^2 + 0.04811 x_3 x_4 (14.0 + x_2) - 5.0 \le 0$$

$$g_5(x): 0.125 - x_1 \le 0$$

$$g_6(x): \delta(x) - \delta_{\max} \le 0$$

$$g_7(x): \mathrm{P} - \mathrm{P}_c(x) \le 0$$

où $\tau(x) = \sqrt{(\tau')^2 + 2\tau'\tau'' \frac{x_2}{2R} + (\tau'')^2}$, $\tau' = \frac{\mathrm{P}}{\sqrt{2} x_1 x_2}$, $\tau'' = \frac{MR}{J}$, $M = P\left(L + \frac{x_2}{2}\right)$,

$R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2}$, $J = 2\left\{\sqrt{2} x_1 x_2 \left[\frac{x_2^2}{12} + \left(\frac{x_1 + x_3}{2}\right)^2\right]\right\}$, $\sigma(x) = \frac{6PL}{x_4 x_3^2}$, $\delta(x) = \frac{4PL^3}{E x_3^3 x_4}$,

$P_c(x) = \frac{4.013E\sqrt{\frac{x_3^2 x_4^6}{36}}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right)$,

avec $P = 6000\,lb$, $L = 14\,in$, $E = 30 \times 10^6\,psi$, $G = 12 \times 10^6\,psi$, $\tau_{\max} = 13,600\,psi$,

$\sigma_{\max} = 30,000\,psi$, $\delta_{\max} = 0.25\,in$, $X = (x_1, x_2, x_3, x_4)^T$, $0.1 \le x_1, x_4 \le 2.0$,

$0.1 \le x_2, x_3 \le 10$.

b) Problème de cuve sous pression : «pressure vessel»

Le problème de cuve sous pression consiste à minimiser le coût total du matériel associé au processus de formage et de soudage d'un réservoir cylindrique, comme illustré en figure IV.5b. Les quatre variables de conception à optimiser sont x1 (l'épaisseur T_S de la coquille), x2 (l'épaisseur T_H de la tête), x3 (le rayon intérieur R) et x4 (la longueur L de la section cylindrique du réservoir, à l'exclusion de la tête). x1 et x2 doivent être des multiples entiers de

0,0625 pouce, qui sont les épaisseurs des tôles d'acier laminées disponibles. Le rayon x3 et la longueur x4 sont des variables continues. Le problème de conception d'une cuve sous pression est un problème mixte avec deux variables discrètes et deux continues, exprimé par :

$$Min\ f\ (x) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3 \qquad \text{(IV.22)}$$

Tel que :

$$g_1(x): -x_1^2 + 0.0193x_3 \le 0$$

$$g_2(x): -x_2 + 0.00954 \le 0$$

$$g_3(x): -\pi x_3^2 x_4 - \frac{4}{3}\pi x_3^3 + 1296000 \le 0$$

$$g_4(x): x_4 - 240 \le 0$$

c) Problème de traction/compression d'une corde : «tension/compression string»

Le problème de traction/compression d'une corde consiste en la minimisation du poids du ressort de traction/compression schématisé dans la figure IV.5c. Le problème est soumis à des contraintes inégalités définies par rapport à la déflexion minimale, le cisaillement, la fréquence de surtension, le diamètre et les variables de conception. Les variables de conception sont le diamètre du fil d (x1), le diamètre moyen de la bobine D (x2), et le nombre de bobines actives N (x3). Le problème est formulé comme suit :

$$Min\ f\ (x) = (x_3 + 2)x_2x_1^2 \qquad \text{(IV.23)}$$

Tel que :

$$g_1(x): 1 - \frac{x_2^3 x_3}{71785x_1^4} \le 0$$

$$g_2(x): \frac{4x_2^2 - x_1x_2}{12566(x_2x_1^3 - x_1^4)} + \frac{1}{5108x_1^2} - 1 \le 0$$

$$g_3(x): 1 - \frac{140.45x_1}{x_2^2x_3} \le 0$$

$$g_4(x): \frac{x_1 + x_2}{1.5} - 1 \le 0$$

d) Problème du *train d'engrenages* : «gear train design»

La conception du train d'engrenages est basée sur la minimisation du rapport d'engrenage du train, comme illustré en Figure IV.5d. Le rapport d'engrenage est défini par :

$gear\ ratio = \frac{n_B n_D}{n_F n_A}$, où n_A, n_B, n_F et n_D représentent les nombres des dents des engrenages A, B, F et D, respectivement.

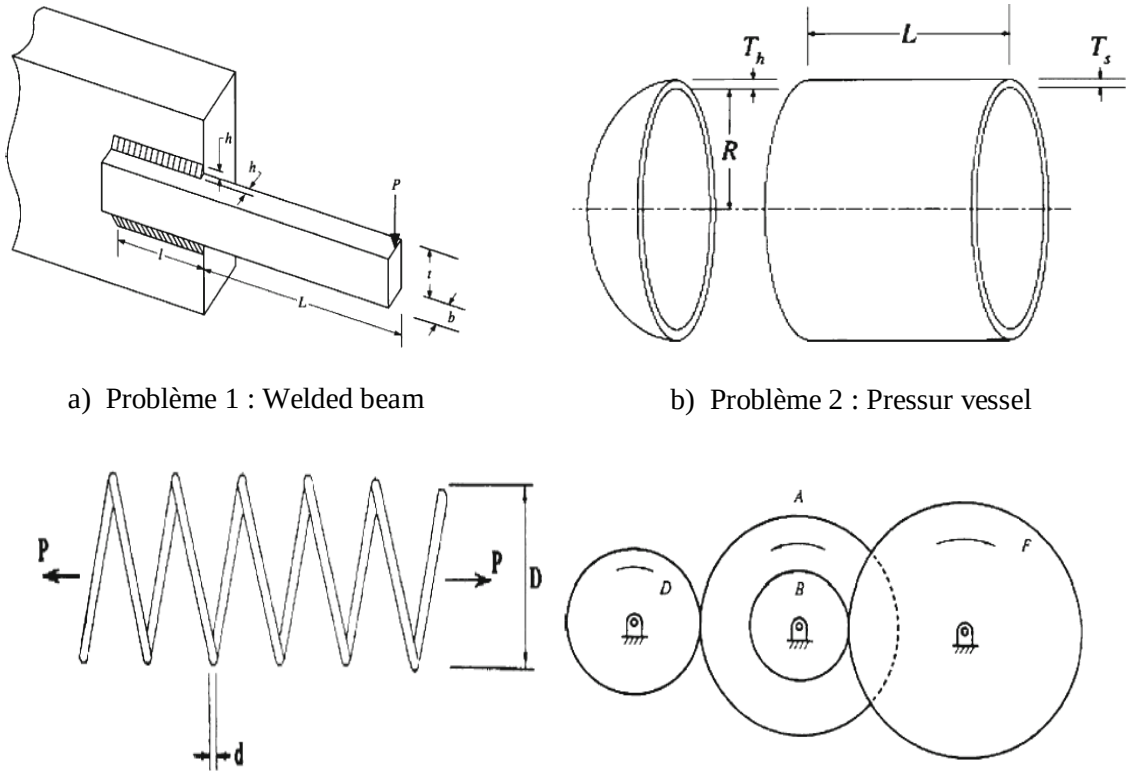
Les variables de conception du problème n_A, n_B, n_D, et n_F seront désignées par x1, x2, x3 et x4, respectivement. Ce sont des entiers compris dans l'intervalle [12, 60]. Ainsi formulé,

le problème du train d'engrenages est un problème discret, dont la formulation est donnée par :

$$Min\ f(x) = \left(\frac{1}{6.931} - \frac{x_3 x_2}{x_1 x_4}\right)^2 \qquad \text{(IV.24)}$$

Tel que :

$12 \le x_i \le 60,\ i = 1,...,4$



a) Problème 1 : Welded beam

b) Problème 2 : Pressur vessel

c) Problème 3 : Ten/comp. string

d) Problème 4 : Gear train

Figure IV.5 : Problèmes de conception mécanique.

IV.5.3 Résultats d'application

La résolution des problèmes de conception mécanique avec contraintes par l'utilisation de l'approche proposée repose sur les mêmes considérations explicitées dans [65] et [123]. Les résultats d'application sont comparés à ceux obtenus par les approches SCA (algorithme d'optimisation basé sur le modèle de société et de civilisation) [125], PSO [126], (μ+λ)-ES (stratégie d'évolution améliorée) [127], UPSOm (algorithme PSO unifié avec mutation) [128], ABC [65] et UABC (algorithme ABC amélioré) [123]. Les paramètres de contrôle utilisés par

les algorithmes ABC, UABC et mCLABC sont indiqués dans le tableau IV.9, où CS représente la taille de la colonie, Max_FEs le nombre maximal d'évaluations de la fonction objectif, et SPP le cycle nécessaire pour exécuter la phase Scout. Comme il peut être constaté, la taille de la colonie définie pour l'algorithme mCLABC est supérieure aux algorithmes comparés. C'est un détail d'ordre computationnel qui a pour objet d'augmenter la diversité des solutions en absence d'adaptation aux cas discret et mixte, comme c'est le cas pour l'algorithme ABC [65] et l'algorithme UABC [123].

Tableau IV.9 : Les paramètres de contrôle des algorithmes ABC, UABC et mCLABC

	ABC	UABC	mCLABC
CS	30	60	80
Max_FEs	30000	15000	15000
SPP	400	100	400
limit	CS*D*5	CS*D*0.5	CS*D*5

Les résultats statistiques obtenus pour les quatre problèmes de conception mécanique étudiés sont représentés dans le tableau IV.10, où sont affichées les valeurs maximales d'évaluation de la fonction objectif de tous les algorithmes comparés.

Tableau IV.10 : Résultats statistiques d'application de l'approche mCLABC proposée aux problèmes de conception mécanique avec contraintes

Problème		**SCA**	**PSO**	**(μ+λ)-ES**	**UPSOm**	**ABC**	**UABC**	**mCLABC**
Welded beam	Meilleure	-	-	1.724852	1.92199	**1.724852**	**1.724852**	**1.724852**
	Moyenne	-	-	1.777692	2.83721	1.741913	**1.724853**	**1.724853**
	Éc.-Type	-	-	8.8e-2	0.682980	3.1e-02	1.7e-06	**1.3e-06**
	Max_FEs	-	-	30,000	100,000	30,000	15,000	15,000
Pressure vessel	Meilleure	6171.00	6059.7143	**6059.701610**	6544.27	6059.714736	6059.714335	6059.852800
	Moyenne	6335.05	6289.92881	6379.938037	9032.55	6245.308144	6192.116211	**6100.555950**
	Éc.-Type	-	3.1e+2	2.1e+2	995.573	2.05e+02	2.04e+02	**80.504**
	Max_FEs	20,000	30,000	30,000	100,000	30,000	15,000	15,000
Ten/comp. string	Meilleure	0.012669	**0.012665**	0.012689	0.0131200	**0.012665**	**0.012665**	**0.012665**
	Moyenne	0.012923	0.012702	0.013165	0.0229478	0.012709	0.012683	**0.012681**
	Éc.-Type	5.9e-4	4.1e-5	3.9e-4	0.00720571	0.012813	3.31e-05	**1.594e-05**
	Max_FEs	25,167	15,000	30,000	100,000	30,000	15,000	15,000
Gear train	Meilleure	-	-	-	**2.700857e-12**	**2.700857e-12**	**2.700857e−12**	**2.700857e-12**
	Moyenne	-	-	-	3.80562e-8	3.641339e-10	**3.500171e−10**	5.786478e-10
	Éc.-Type	-	-	-	1.09631e-7	5.525811e-10	**4.690420e−10**	6.965663e-10
	Max_FEs	-	-	-	100,000	60	3,000	3,000

D'après le tableau IV.10, il y a lieu d'abord de constater que les algorithmes à essaims d'abeilles ABC, UABC et mCLABC sont supérieurs à tous les autres algorithmes comparés. Le modèle mCLABC adapté aux problèmes d'optimisation avec contraintes affiche des résultats satisfaisants dans la grande majorité des cas étudiés. L'amélioration apportée sur l'erreur moyenne et sa déviation est significative pour le problème de cuve sous pression.

IV.6 Conception d'un compensateur de non linéarité pour le transducteur LVDT

Le transducteur de déplacement inductif à transformateur différentiel (LVDT) est un instrument de mesure de déplacement très utilisé dans les chaines de mesure et d'acquisition industrielles. Compte tenu de la conception du transducteur, le LVDT présente une caractéristique de mesure non linéaire, dont le degré de non-linéarité varie en fonction de la nature du transducteur et des conditions d'exploitation. Pour une mesure fiable et précise, la compensation de la non linéarité du transducteur s'impose, ce qui nécessite la mise en place d'une solution adaptée à l'application en question. Plusieurs travaux sont axés sur cette problématique et les méthodes utilisées pour la résoudre sont multiples et variées, certaines mêmes formulées en tant que problèmes d'optimisation non linéaire [129] [130] [131] [132]. Parmi ces méthodes de compensation, nous pourrions citer les algorithmes évolutionnaires [131] et les réseaux de neurones de type FLANN [129] [133]. Dans la suite, nous résumons les résultats d'application de la nouvelle stratégie mCLABC au problème de compensation de non-linéarité du LVDT.

IV.6.1 Le capteur inductif linéaire LVDT

Le capteur de déplacement inductif LVDT (Figure IV.6) se compose d'un enroulement (bobine) primaire (transformateur statique) et de deux enroulements secondaires. Les deux enroulements secondaires sont reliés différemment pour fournir la sortie. Ils sont situés sur les deux côtés de l'enroulement primaire. Ces deux enroulements de sortie sont connectés en opposition afin de produire une sortie zéro à la position médiane de l'armature. A l'intérieur des bobines, une armature (noyau) ferromagnétique se déplace dans une direction axiale. Le noyau ferromagnétique fournit un cheminement permettant de relier le flux magnétique généré dans la bobine primaire, lorsque l'enroulement primaire est relié à une alimentation alternative, vers les bobines secondaires. La position du noyau à l'intérieur de la bobine contrôle le débit de flux lié à chaque enroulement secondaire. Les enroulements secondaires A et B sont connectés en opposition de série, de sorte que les deux tensions V_A et V_B sont en opposition de phase et la tension de sortie du capteur est V_A-V_B. Lorsque le noyau se déplace dans une direction, la tension dans la bobine secondaire augmente tandis que l'autre bobine subit une réduction de tension complémentaire. L'effet net est une tension de sortie au niveau du capteur qui est proportionnelle au déplacement et à la position du noyau.

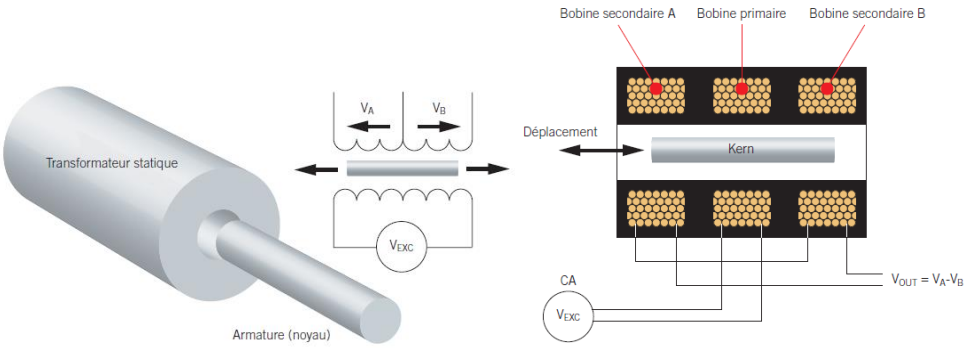


Figure IV.6 : Schéma simplifié d'un transducteur LVDT.

IV.6.2 Compensation de non linéarité par réseau FLANN-mCLABC

Parmi les concepts étudiés dans la littérature, on recense ceux qui se basent sur le calcul de modèles inverses à partir de la caractéristique non-linéaire du transducteur LVDT. Pour le calcul de ce modèle, différentes techniques de l'intelligence artificielle ont été employées à l'exemple des réseaux de neurones artificiels, les systèmes flous, les algorithmes génétiques, …etc. L'approche de conception du compensateur de non-linéarité du LVDT que nous proposons d'étudier repose sur le modèle neuronal FLANN (Functional Link Artificial Neural Network) optimisé par l'algorithme mCLABC. Il sera désigné par FLANN-mCLABC dans la suite de cette description.

La figure IV.7 illustre la structure générale du réseau FLANN. Il s'agit d'un réseau de neurones plat, sans couches cachées. La tension à la sortie du LVDT (v), qui est non linéaire, est fournie à l'entrée du modèle FLANN. Il est soumis à des expansions fonctionnelles utilisant des séries mathématiques (trigonométriques, séries de puissance, tenseur, produit extérieur, etc). Ces liens fonctionnels agissent comme un modèle de fonctions linéairement indépendantes, évaluées avec ce modèle en tant qu'argument. Dans cette étude, nous utiliserons des expansions trigonométriques, car elles permettent une meilleure compensation de non-linéarité par rapport aux autres séries mathématiques [134]. Ces expansions sont multipliées par un ensemble de poids neuronaux, et finalement sommées pour produire la sortie du modèle inverse. Le signal de sortie est comparé au signal souhaité (signal de commande d'actionneur de déplacement) pour en déduire le signal d'erreur. Les poids sont mis à jour en minimisant l'erreur quadratique moyenne (MSE). Le processus de calcul est répété jusqu'à ce que la valeur de MSE atteigne un seuil minimum, au-delà duquel il n'améliore pas les estimations. La double combinaison du LVDT et du FLANN-mCLABC représente un capteur linéaire avec une plage de linéarité améliorée.

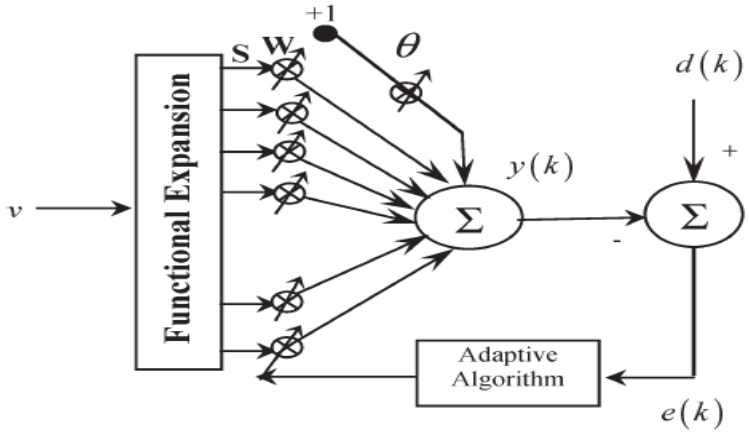


Figure IV.7 : Structure du modèle FLANN optimisé par un algorithme évolutionnaire.

Considérons à présent le problème d'apprentissage du réseau FLANN. Soit V le vecteur d'entrée de dimension N. Chaque élément d'entrée subit une expansion trigonométrique non linéaire pour former des éléments P de telle sorte que la matrice résultante ait la dimension $N \times P$. Étant donné que le n[ème] élément d'entrée est v_n, $1 \leq n \leq N$, l'expansion fonctionnelle est effectuée comme suit [129] :

$$s_i = \begin{cases} v_n, & i = 1 \\ sin(l\pi v_n), & i > 1, \ \ i \ est \ paire \\ cos(l\pi v_n), & i > 1, \ \ i \ est \ impaire \end{cases} \quad \text{(IV.25)}$$

où $l = 1,2,\dots,P/2,$ et $i = 1,2,\dots,P+1.$ Dans l'extension proposée, le nombre total d'extensions P est toujours un nombre pair. En notation matricielle, les éléments expansés du vecteur d'entrée V sont désignés par S de taille $N \times (P+1)$. Une valeur d'unité supplémentaire (pour le biais) est capitonnée avec la matrice S, et donc, la dimension de la matrice S devient $N \times Q$, où $Q = (P+2)$.

Soit le vecteur de poids W composé de Q éléments. La sortie y est donnée par :

$$y = \sum_{i=1}^{Q} W_i S_i \quad \text{(IV.26)}$$

En notation matricielle, la sortie peut être réécrite sous la forme :

$$Y = S.W^T \quad \text{(IV.27)}$$

À l'itération k, le signal d'erreur entre le modèle estimé et le modèle désiré $e(k)$ est donné par :

$$e(k) = d(k) - y(k) \quad \text{(IV.28)}$$

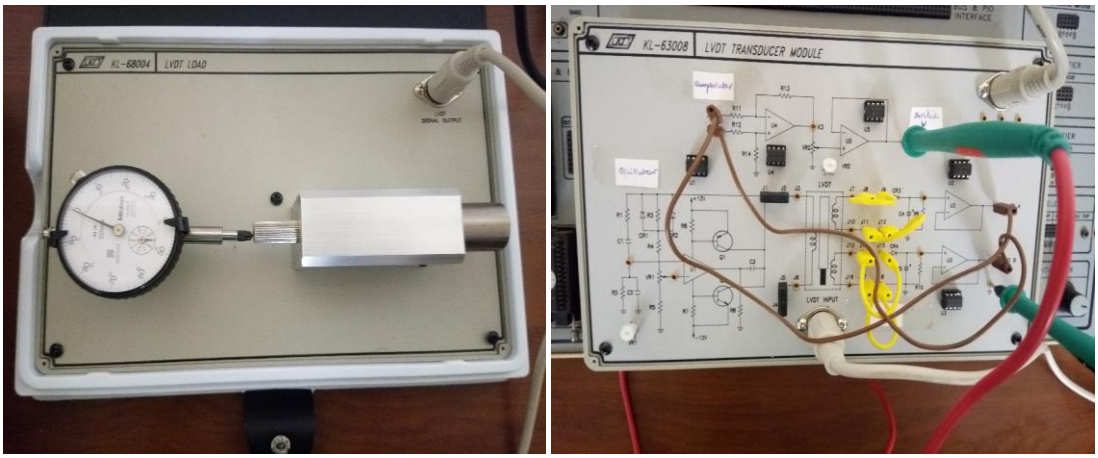
où $d(k)$ est le signal souhaité qui est identique au signal de commande transmis à l'actionneur de déplacement. En se basant sur l'erreur, on définit la fonction objectif à minimiser par :

$$\epsilon(k) = \frac{1}{2}\sum_{j \in P} e_j^2(k) \quad \text{(IV.29)}$$

Afin de réaliser un réseau FLANN-mCLABC, l'approche mCLABC est utilisée pour ajuster le vecteur des poids W, de sorte qu'à chaque mise à jour de solutions, la sortie produite par le réseau FLANN-mCLABC est comparée avec la sortie désirée et l'erreur d'estimation du modèle inverse est donc évaluée. À la fin du processus d'optimisation par mCLABC, le réseau FLANN fournit la caractéristique inverse du transducteur LVDT qui sera utilisée pour déduire la caractéristique linéarisée.

IV.6.3. Résultats d'application

Pour évaluer la performance de l'approche FLANN-mCLABC, deux différentes bases de données expérimentales sont utilisées. La première base LVDT-1 est reprise de la référence [131], alors que la deuxième base de données LVDT-2 est prélevée sur un transducteur du type KL-68004 disponible au sein de notre laboratoire (Figure IV.8). Les données expérimentales utilisées dans la phase de conception sont indiquées en Annexe B.



(a) Le LVDT KL-68004 (b) Le circuit de conditionnement

Figure IV.8 : Le LVDT du laboratoire (KL-68004) et son circuit de conditionnement.

Pour cette étude de compensation de non linéarité du LVDT, le réseau FLANN est traité moyennant l'algorithme ABC, l'algorithme gABC [55] et l'algorithme mCLABC. L'apprentissage à base de ces méthodes est effectué séparément en considérant une architecture de réseau composé de 51 poids à optimiser [129]. Un ensemble de 30 essais est considéré pour chaque problème. Pour une comparaison fiable, les méthodes comparées sont utilisées avec les mêmes paramètres de contrôle communs, tels que reporté dans [129]. La taille de la population est 100, le cycle maximal est fixé à 1000, et la valeur $limit=SN\times D$. Les résultats comparatifs des compensateurs FLANN-mCLABC, FLANN-ABC, FLANN-gABC,

le compensateur basé sur le réseau ELM (Extreme Learning Machine) [129], le compensateur basé sur l'algorithme d'évolution différentielle DE [129], et le compensateur basé sur le réseau ANN optimisé par les algorithmes génétiques (GA-ANN) [129] sont dressés dans les tableaux IV.11 et IV.12.

La figure IV.9 illustre l'évolution de l'erreur quadratique moyenne obtenue pour les modèles FLANN-mCLABC, FLANN-ABCet FLANN-gABC.

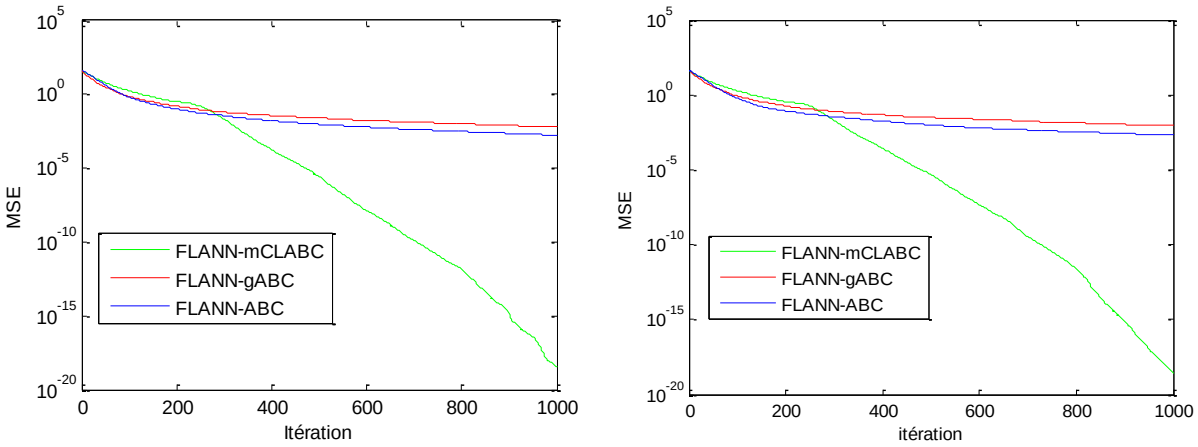
Pour mieux appréhender les différences entre les approches de compensation de non-linéarité étudiées, nous avons convenu de reproduire les tracés des caractéristiques du LVDT-1 et LVDT-2 avant et après compensation. Les caractéristiques obtenues par le compensateur ELM, l'algorithme DE et le modèle GA-ANN reprises de la référence [131] sont montrées en figure IV.10. Les caractéristiques obtenues par les modèles FLANN-mCLABC, FLANN-ABCet FLANN-gABC sont montrées en figures IV.11-IV.13.

Tableau IV.11 : Comparaison des performances du modèle FLANN-mCLABC avec d'autres approches avancées sur le problème de compensation du LVDT-1

Méthode	**Paramètres de la méthode**	**MSE (Moyenne)**
ELM	Sinus Sigmoïde	0.087 0.0088
Algorithme DE	F=0.6 CR=0.3 NP=100	0.0084
GA-ANN	NP=80	0.0066
FLANN-ABC	NP=100	0.00168781 (0.000739866)
FLANN-gABC	NP=100	0.00624393 (0.00587724)
FLANN-mCLABC	NP=100	**3.3029e-19** **(1.7367e-18)**

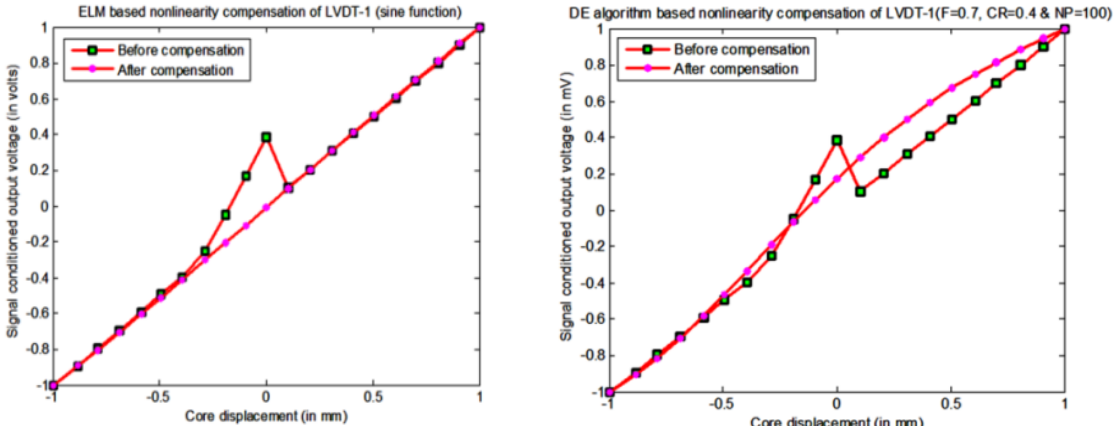
Tableau IV. 12 : Comparaison des performances du modèle FLANN-mCLABC avec d'autres approches avancées sur le problème de compensation du LVDT-2

Méthode	**Taille de la colonie**	**MSE (Moyenne)**
FLANN-ABC	NP=100	0.00200337 (0.000976429)
FLANN-gABC	NP=100	0.00907795 (0.012883)
FLANN-mCLABC	NP=100	**2.77423e-19** **(7.89832e-19)**

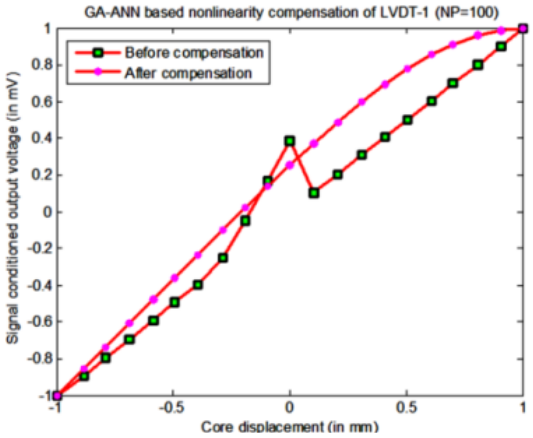


LVDT-1 LVDT-2

Figure IV.9 : Les courbes de convergence des compensateurs FLANN-ABC, FLANN-gABC et FLANN-mCLABC développés pour les transducteurs LVDT-1 et LVDT-2.

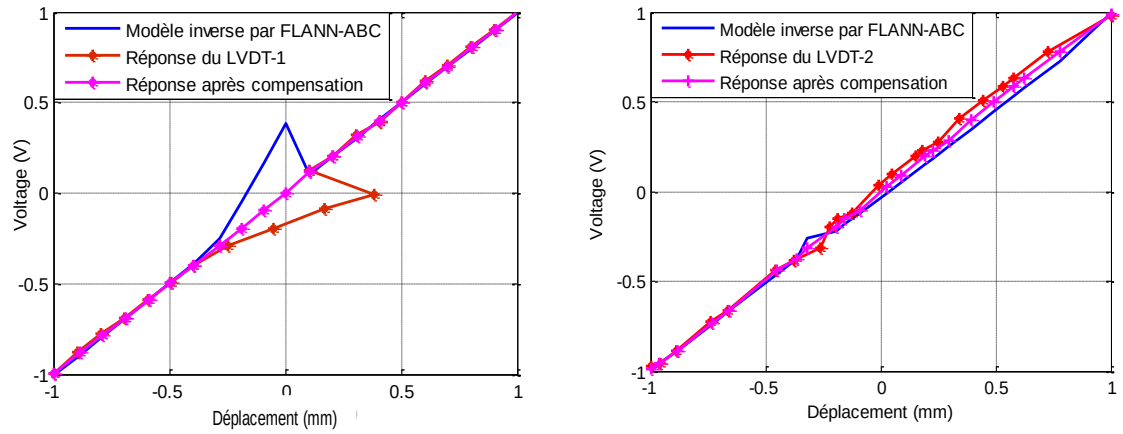


ELM (LVDT-1) DE (LDVT-1)



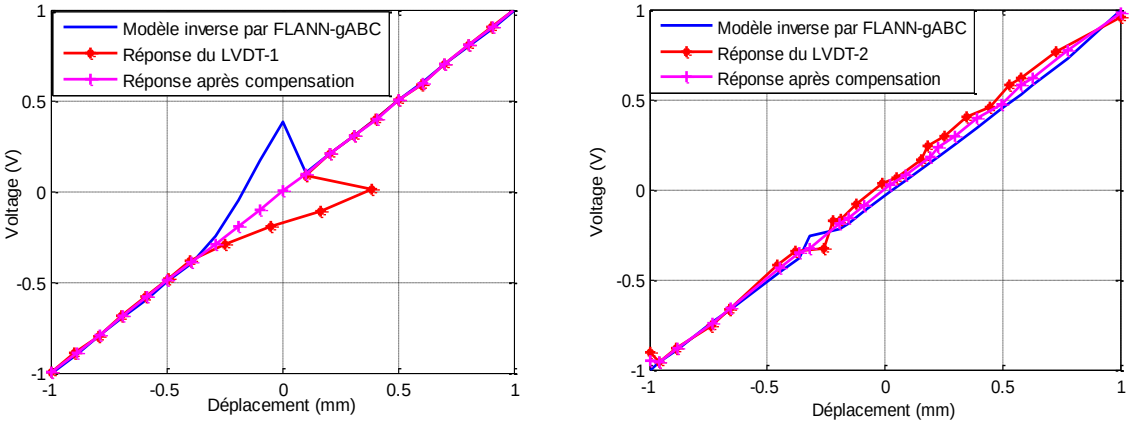
GA-ANN (LVDT-1)

Figure IV.10 : Compensation de la non-linéarité du LVDT-1 par le modèle ELM (sinus), l'algorithme DE et le modèle GA-ANN [131].



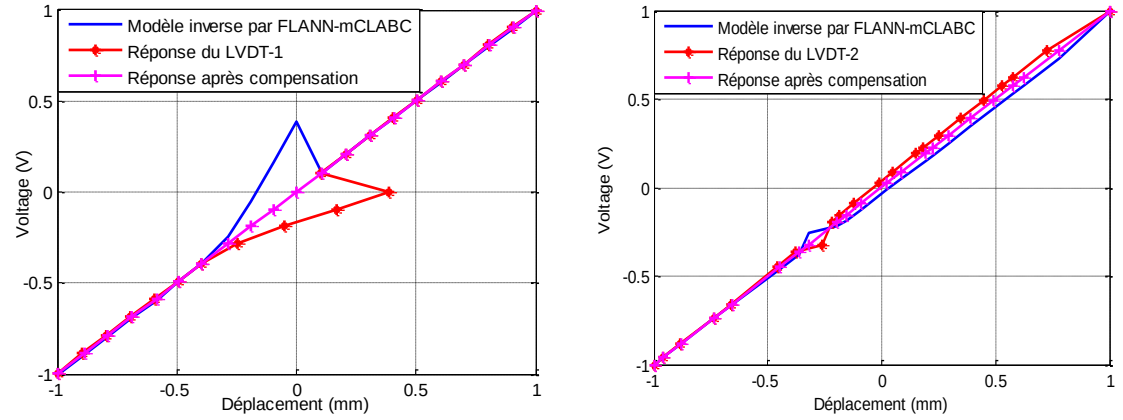
FLANN-ABC (LVDT-1) FLANN-ABC (LVDT-2)

Figure IV.11 : Compensation de la non-linéarité du LVDT-1 et du LVDT-2 par le modèle FLANN-ABC.



FLANN-gABC (LVDT-1) FLANN-gABC (LVDT-2)

Figure IV.12 : Compensation de la non-linéarité du LVDT-1 et du LVDT-2 par le modèle FLANN-gABC.



FLANN-mCLABC (LVDT-1) FLANN-mCLABC (LVDT-2)

Figure IV.13 : Compensation de la non-linéarité du LVDT-1 et du LVDT-2 par le modèle FLANN-mCLABC.

D'après les résultats obtenus, la performance du modèle FLANN-mCLABC proposé est nettement supérieure avec une erreur d'estimation moyenne très proche de zéro pour les deux bases de données LVDT-1 et LVDT-2.

IV.7 Optimisation du contrôle PID

Le reparamétrage du contrôleur PID est l'une des situations les plus fréquentes auxquelles font face les automaticiens en industrie. L'objectif étant de garantir le bon fonctionnement des boucles de régulation à travers une stabilisation adéquate de tout le système de contrôle. Les opérateurs en charge des systèmes automatisés sont souvent appelés à gérer des procédures d'ajustement des paramètres des contrôleurs installés selon des méthodes empiriques assez maîtrisés. L'apport que pourrait apporter toute méthode non conventionnelle est souvent le résultat de manipulation de mécanismes basés sur des approches avancées, telles que la logique floue, les réseaux de neurones, les métaheuristiques et bien d'autres. Un paramétrage à base de métaheuristiques est justement perçu comme étant un problème d'optimisation défini sur les trois actions de réglage, en l'occurrence l'action proportionnelle, intégrale et dérivée. La résolution du problème est tributaire de la méthode utilisée. Dans ce contexte, nous proposons dans la suite d'examiner l'apport que pourrait apporter la nouvelle approche mCLABC en matière de paramétrage optimale des boucles de réglage PID.

IV.7.1 Le contrôleur PID

La loi de la commande PID s'exprime en temps continu par :

$$\mathrm{u}(t) = k_p \left[e(t) + \frac{1}{T_i} \int_0^t e(\tau) d\tau + T_d \frac{de(t)}{dt} \right] \tag{IV.30}$$

où k_p, T_i et T_d sont, respectivement, les actions proportionnelle, intégrale et dérivée. D'une manière équivalente, la fonction de transfert associée peut être écrite sous la forme :

$$G_c(s) = k_p + \frac{k_i}{s} + k_d s \tag{IV.31}$$

où k_p, k_i et k_d sont respectivement les gains proportionnel, intégral et dérivé.

Le paramétrage optimal du contrôleur PID est explicité par rapport aux trois gains de réglage k_p, k_i et k_d. Une méthode de résolution à base de métaheuristiques poserait le problème comme étant un problème de recherche globale dont le principe général de calcul est schématisé en figure IV.14.

IV.7.2 Procédure d'optimisation et résultats

Le paramétrage du contrôleur PID à base de métaheuristique repose sur le principe fonctionnel schématisé par la figure IV. 14.

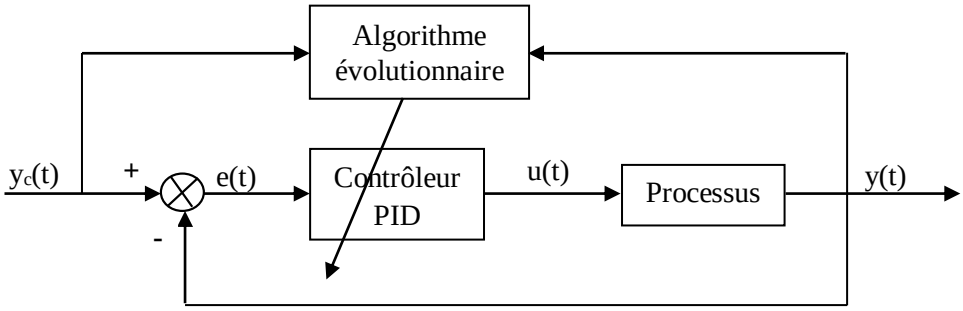


Figure IV.14 : Principe de paramétrage d'un contrôleur PID à base de métaheuristique.

La recherche des paramètres de contrôle optimaux s'effectue moyennant le modèle mCLABC proposé. Pour ce faire, nous considérions le problème de minimisation de l'erreur absolue de réglage (IAE) qui s'exprime par :

$$F(t) = \begin{cases} \int_0^\infty (w_1|e(t)| + w_2|ey(t)|)dt, & si\ ey(t) < 0 \\ \int_0^\infty (w_1|e(t)|)dt & ailleurs \end{cases} \quad \text{(IV.32)}$$

où $ey(t) = y(t) - y(t-1)$ et w_1, w_2 désignent les poids de pondération.

Les performances des réglages par les différentes méthodes comparées sont observées sur les réponses indicielles des processus contrôlés et évaluées en termes d'erreur de réglage, de temps de montée (t_r), de temps de réponse (t_s) et de dépassement (PO). Trois processus de référence suggérés par Aström dans [135] sont utilisés. Les fonctions de transfert des trois processus sont données par les équations (IV.33)- (IV.35) :

$$G_1(s) = \frac{1}{(1+s)(1+\alpha s)(1+\alpha^2 s)(1+\alpha^3 s)} \qquad \alpha = 0.5 \qquad \text{(IV.33)}$$

$$G_2(s) = \frac{1}{(1+Ts)^2} e^{-s} \qquad T = 10 \qquad \text{(IV.34)}$$

$$G_3(s) = \frac{\omega_0^2}{(s+1)(s^2 + 2\xi\omega_0 s + \omega_0^2)} \qquad \omega_0 = 1,\ \xi = 0.1 \qquad \text{(IV.35)}$$

Les valeurs des poids et les plages des gains k_p, k_i et k_d utilisées dans la simulation sont indiquées dans le tableau IV.13.

L'étude comparative est menée par rapport à l'algorithme de recherche harmonique HS, l'algorithme de chauve-souris BA, l'algorithme de chauve-souris amélioré IBA, et l'algorithme ABC, dont les résultats d'application au problème de paramétrage du contrôleur PID sont repris de la référence [41]. Pour une comparaison fiable, les méthodes comparées sont considérés avec les mêmes paramètres de contrôle communs. Un ensemble de 20 essais est considéré pour chaque problème, le nombre maximal d'évaluations est fixé à 1000. Pour les approches ABC et mCLABC, la taille de la colonie est fixée à 20 et le paramètre *"limit"* est égal à 30.

Tableau IV.13 : Poids et plages de variation des paramètres de réglage.

	ω_1	ω_2	Plage K_P	Plage K_i	Plage K_d
Modèle 1	1	100	[0, 10]	[0, 10]	[0, 10]
Modèle 2	1	0	[0, 20]	[0, 5]	[0, 50]
Modèle 3	1	100	[0, 2]	[0, 2]	[0, 2]

Les résultats comparatifs obtenus sont représentés dans le tableau IV.14 en termes d'erreur moyenne (Mean) et d'écart-type (Std) évalués sur les valeurs du minimum global calculés sur 10 essais. Les performances de réglage résultantes sont indiquées dans les tableaux IV.15, IV.16 et IV.17 pour la meilleure, la moyenne et la mauvaise solution, respectivement. Pour l'ensemble de ces résultats, nous illustrons également dans les figures IV.15-IV.20 les réponses indicielles associées à la meilleure et la mauvaise solution, obtenues pour les trois modèles de processus considérés.

D'abord, il y a lieu de constater que les résultats obtenus permettent une meilleure visibilité de l'intérêt d'utiliser les méthodes évolutionnaires dans les problèmes de paramétrage optimal des boucles de réglage PID. A ce titre, l'approche proposée enregistre les meilleures valeurs d'erreurs pour tous les modèles de processus étudiés. Il en résulte alors selon le cas des performances différentes sur les réponses indicielles que le modèle mCLABC arrive à gérer de manière optimale.

Tableau IV.14 : Résultats comparatifs des erreurs moyennes et des écart-types obtenus par les algorithmes ZN, HS, BA, IBA, ABC et mCLABC

	Methode	**ZN**	**HS**	**BA**	**IBA**	**ABC**	**mCLABC**
Processus (1)	Moyenne	104.6956	68.0529	67.3781	67.4198	67.4704	**67.2982**
	Éc.-Type	0	0.8671	0.0614	0.1132	0.1924	**0.0419**
Processus (2)	Moyenne	916.4172	226.1694	237.7946	226.2960	217.9816	**217.3790**
	Éc.-Type	0	7.8657	16.9276	8.3822	0.7717	**0.1377**
Processus (3)	Moyenne	1152.2	187.2940	191.0669	190.7595	186.7577	**186.5530**
	Éc.-Type	0	1.3330	1.9050	1.0737	0.8064	**0.4858**

Tableau IV.15 : Performances du réglage PID obtenues pour la meilleure solution du problème d'optimisation.

	Methode	K_p	K_i	K_d	Erreur	PO	t_r	t_s
Processus (1)	ZN	3.9706	3.5749	1.1026	104.6956	37.5733	**0.62**	3.26
	HS	2.8206	1.8022	1.4330	67.2700	1.4962	0.81	0.99
	IBA	2.7852	1.7873	1.4157	67.2901	**1.1671**	0.82	1.00
	ABC	2.8458	1.8278	1.4535	67.2896	1.7586	0.80	**0.98**
	mCLABC	2.8218	1.7993	1.4321	**67.2698**	1.4963	0.66	0.99
Processus (2)	ZN	12.1599	1.7167	21.5328	916.4172	55.4210	3.14	39.51
	HS	9.9104	0.4955	49.6503	217.9951	3.7729	2.36	3.40
	IBA	9.8857	0.4919	49.8251	218.4367	**3.6894**	2.35	3.40
	ABC	10.0158	0.5003	50.0000	217.4098	4.0968	2.32	**3.37**
	mCLABC	9.9950	0.4997	50.0000	**217.3190**	4.0315	**1.89**	**3.37**
Processus (3)	ZN	0.2588	0.0902	0.1856	1152.2	**1.1368**	31.5500	36.72
	HS	0.2702	1.3582	1.3701	185.9780	21.9206	1.52	4.46
	IBA	0.3000	1.5748	1.5856	188.2611	24.8456	1.37	5.79
	ABC	0.2848	1.4186	1.4334	186.1519	22.7805	1.47	**4.34**
	mCLABC	0.2688	1.3510	1.3629	**186.1320**	21.8094	**1.31**	4.48

Tableau IV.16 : Performances du réglage PID obtenues pour la moyenne solution du problème d'optimisation.

	Methode	K_p	K_i	K_d	Erreur	PO	t_r	t_s
	HS	2.7637	1.8491	1.5404	67.5536	2.4726	0.75	**0.93**
Processus	IBA	2.7781	1.7439	1.3604	67.3630	**1.2937**	0.83	1.02
(1)	ABC	2.7781	1.8275	1.4309	67.3434	1.3709	0.82	1.00
	mCLABC	2.8230	1.8022	1.4329	**67.2700**	1.5279	**0.66**	0.99
	HS	9.2736	0.4666	48.4436	223.6305	**1.8263**	2.53	3.58
Processus	IBA	9.4173	0.4723	47.0724	223.6276	2.3704	2.59	3.65
(2)	ABC	10.0575	0.5036	49.9966	217.6687	4.2286	2.32	**3.37**
	mCLABC	9.9951	0.4997	49.9999	**217.319**	4.0317	**1.89**	**3.37**
	HS	0.2390	1.1830	1.1932	186.5621	**19.1465**	1.68	4.80
Processus	IBA	0.3885	2.0000	2.0000	191.2304	30.2721	**1.16**	5.56
(3)	ABC	0.2910	1.4490	1.4535	186.3248	23.6555	1.45	6.06
	mCLABC	0.2586	1.2937	1.3054	**186.1770**	20.9180	1.34	**4.58**

Tableau IV.17 : Performances du réglage PID obtenues pour la mauvaise solution du problème d'optimisation.

	Methode	K_p	K_i	K_d	Erreur	PO	t_r	t_s
	HS	3.2817	1.9190	1.8233	69.8389	4.2752	0.65	2.20
Processus	IBA	2.9328	1.8617	1.4587	67.6010	3.0331	0.77	0.95
(1)	ABC	3.0455	1.8886	1.5853	67.8612	3.1383	0.73	**0.91**
	mCLABC	2.9220	1.8294	1.5042	**67.3964**	**2.0702**	**0.64**	0.95
	HS	11.4317	0.5601	49.5014	242.9424	8.2906	2.20	6.51
Processus	IBA	11.5841	0.5960	50.0000	244.2850	8.8584	2.16	6.63
(2)	ABC	10.1405	0.5016	50.0000	219.6083	4.4450	2.31	**3.36**
	mCLABC	9.9043	0.4947	49.9767	**217.7460**	**3.7552**	**1.9**	3.38
	HS	0.3562	1.8363	1.8460	189.1992	28.2260	1.23	5.66
Processus	IBA	0.3834	2.0000	2.0000	191.2984	30.1697	**1.16**	5.56
(3)	ABC	0.3333	1.6804	1.6901	188.2179	26.5106	1.30	5.82
	mCLABC	0.2267	1.1165	1.1247	**187.4680**	**18.0595**	1.49	**4.95**

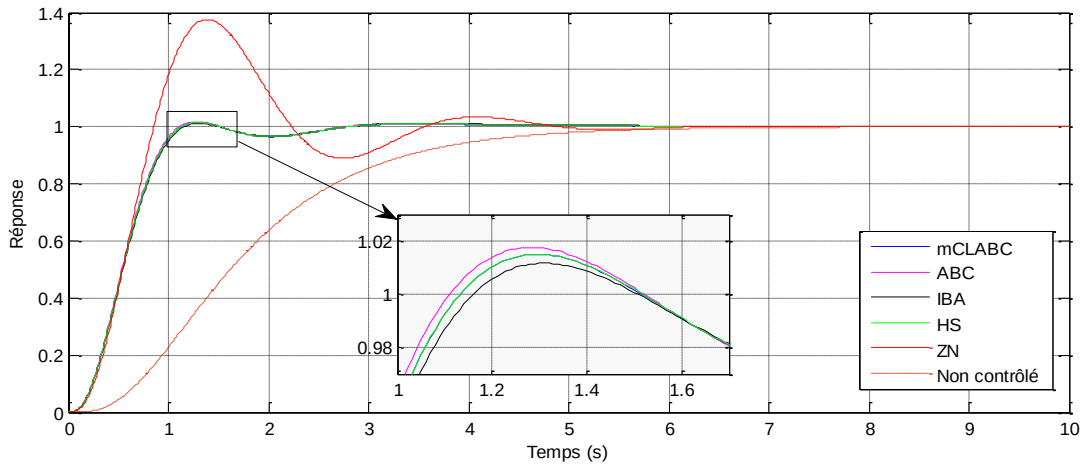


Figure IV.15 : La réponse indicielle du processus 1 associée à la meilleure solution obtenue par les méthodes ZN, HS, IBA, ABC et mCLABC.

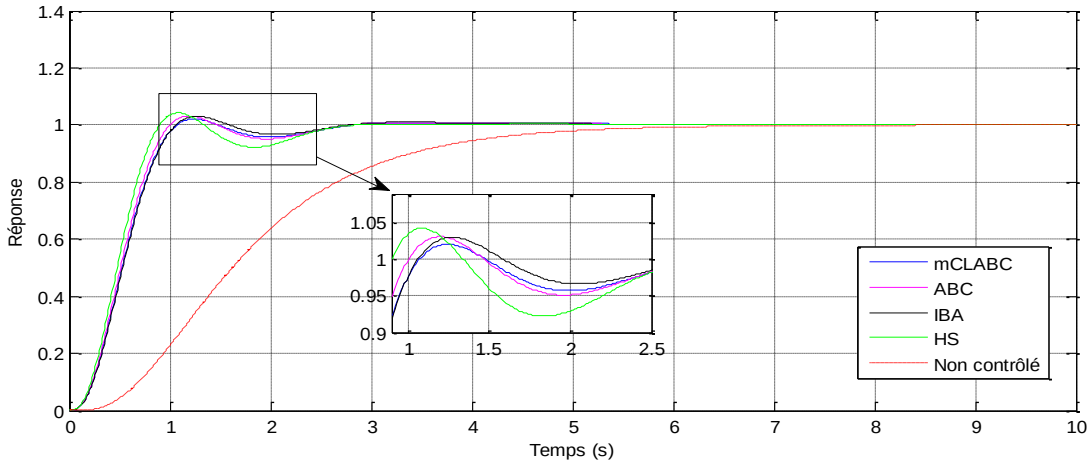


Figure IV.16 : La réponse indicielle du processus 1 associée à la mauvaise solution obtenue par les méthodes HS, IBA, ABC et mCLABC.

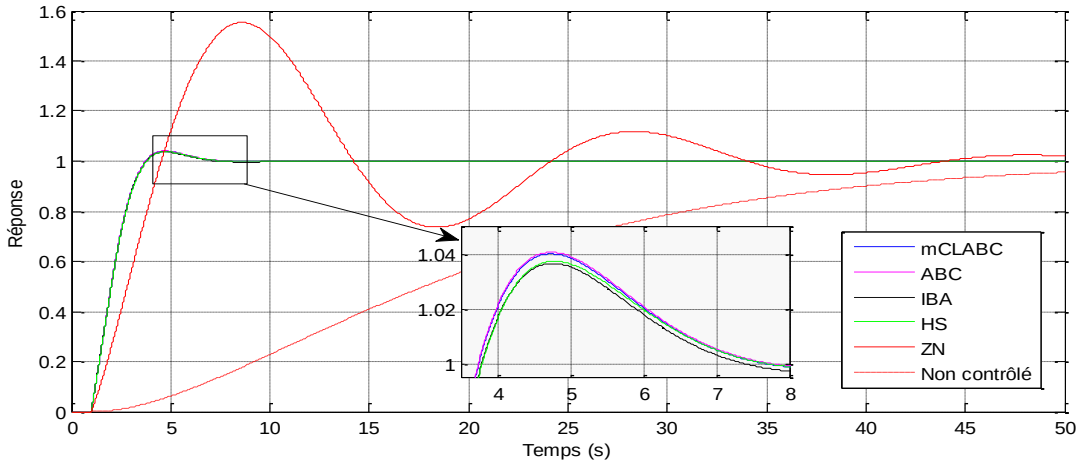


Figure IV.17 : La réponse indicielle du processus 2 associée à la meilleure solution obtenue par les méthodes ZN, HS, IBA, ABC et mCLABC.

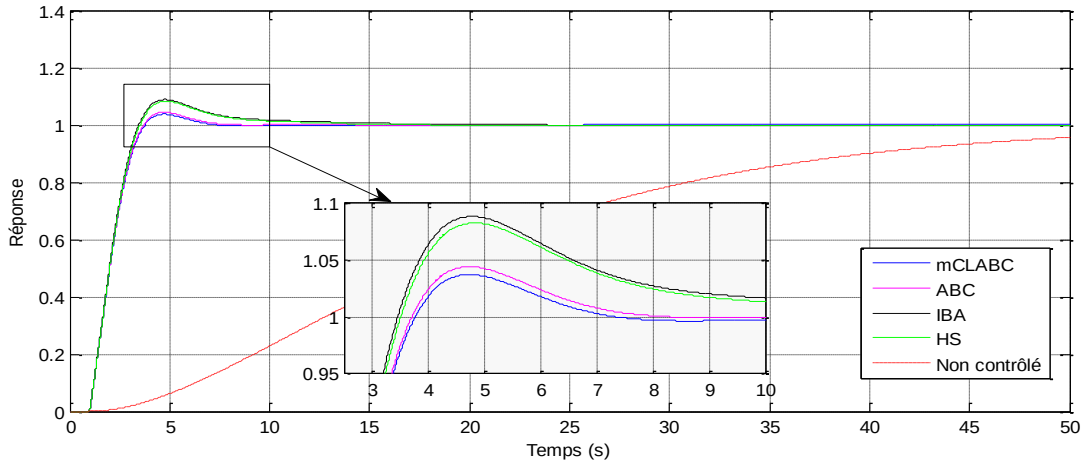


Figure IV.18 : La réponse indicielle du processus 2 associée à la mauvaise solution obtenue par les méthodes HS, IBA, ABC et mCLABC.

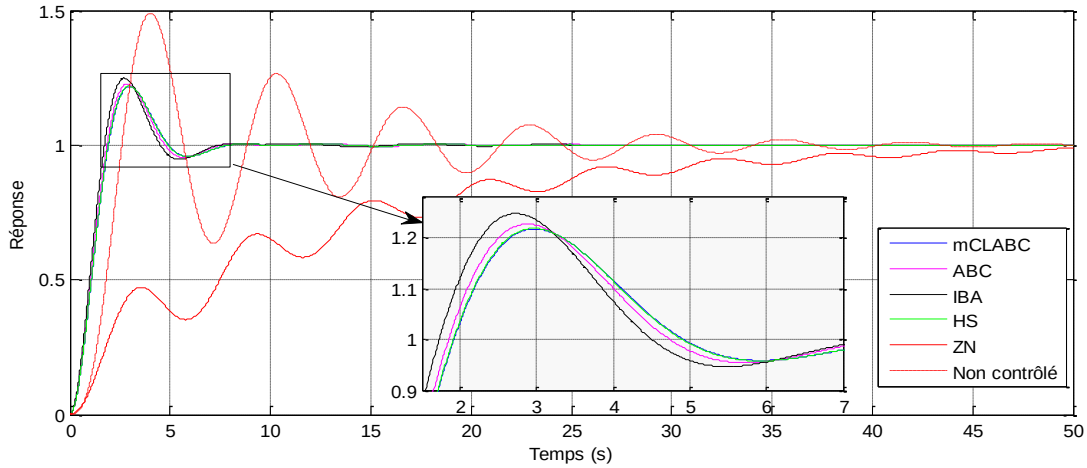


Figure IV.19 : La réponse indicielle du processus 3 associée à la meilleure solution obtenue par les méthodes ZN, HS, IBA, ABC et mCLABC.

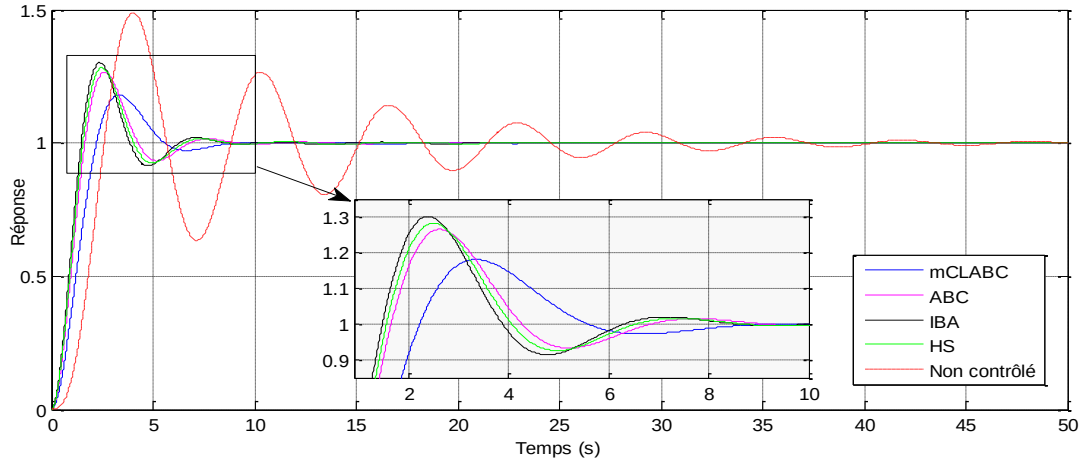


Figure IV.20 : La réponse indicielle du processus 3 associée à la mauvaise solution obtenue par les méthodes HS, IBA, ABC et mCLABC.

IV.8 Conclusion

L'étude de validation de l'approche d'optimisation par essaims d'abeilles à apprentissage coopératif sur des problèmes scientifiques constitue l'objet principal de ce quatrième chapitre. C'est une étape fondamentale dans le processus de développement de nouvelles approches, que nous avions bien voulu intégré dans ce travail de thèse, et ce à partir des conclusions établies sur la base des analyses intensives de performance et de robustesse présentées dans le Chapitre III. Pour ce faire, six différentes applications relevant du domaine des systèmes intelligents et de l'ingénierie des systèmes ont été examinées. Il s'agit tout précisément de problèmes scientifiques posés en tant que problèmes d'optimisation difficile, que nous proposons de résoudre moyennant la nouvelle approche d'optimisation.

Les applications appréhendées concernent l'identification des systèmes neuro-flous, la conception des classifieurs neuronaux, la classification de données non supervisée, la conception mécanique sous contraintes, la compensation de la non-linéarité d'un transducteur inductif LVDT et l'optimisation du contrôle PID. Certaines de ces études de cas s'appuient sur des données expérimentales réelles ou des modèles de conception physiques. La faisabilité de l'approche proposée en tant que nouvel outil d'optimisation est l'importante conclusion qui en ressort de cette étude de validation menée sur la base de résultats comparatifs repris de travaux récents traitant les mêmes problèmes avec des méthodes de résolution avancées. La stratégie d'optimisation proposée s'est montrée imposante dans la plupart des cas d'application étudiés, ce qui conforte par excellence la contribution de la présente thèse.

Conclusion générale

De nombreux problèmes scientifiques et techniques relevant de domaines d'application diverses sont formulés sur la base de concepts d'optimalité, nécessitant par conséquent l'intervention d'une ou plusieurs méthodes d'optimisation pour les résoudre. La recherche d'états optimaux est une démarche déterminante pour ce type de problèmes, dont l'appréhension préalable est d'une importance majeure pour évaluer la méthode d'optimisation à utiliser. La thématique abordée dans cette thèse traite justement de la problématique d'optimisation avec l'objectif d'apporter quelques contributions à l'optimisation par méthodes approchées dédiées aux problèmes difficiles de grande dimension. Pour cette classe de méthodes, nous nous sommes particulièrement intéressés aux métaheuristiques de l'intelligence en essaim qui renferment une multitude d'approches de recherche globale à base de population de solutions.

Dans ce sens, notre intérêt a été tout précisément porté sur l'optimisation par colonies d'abeilles artificielles (ABC), qui s'est vue développer une panoplie de variantes appliquées avec succès à différents problèmes scientifiques. Le principal objectif que nous nous sommes d'abord fixés concerne le développement d'une nouvelle stratégie d'optimisation intelligente qui s'inspire du modèle ABC, mais qui soit totalement différente sur le plan de la caractérisation comportementale des essaims d'abeilles pendant le processus de fourragement collectif. Pour ce faire, il a été question d'introduire des modifications structurelle et paramétrique au modèle ABC de base, de manière à modéliser au mieux les concepts comportementaux définis dans la démarche de conception.

L'intégration de ces concepts à la plateforme algorithmique de base du modèle ABC est passée d'abord par la modélisation de trois principaux processus comportementaux, en l'occurrence le processus du fourragement coopératif, le processus de recherche à règles multiples et le processus d'altération des sources de nourriture. Au vu de ces considérations de conception, une nouvelle structure algorithmique est avancée, répondant à la désignation CLABC « Optimisation par colonies d'abeilles artificielles à apprentissage coopératif ».

Cette étape de développement est ensuite confortée par une analyse de performance illustrant des résultats comparatifs de la stratégie proposée au modèle ABC de base et à ses principales variantes, ainsi qu'à d'autres métaheuristiques avancées rapportées dans des travaux récents. Un large panel de problèmes numériques est considéré dans les différents

tests de validation, à l'exemple des fonctions numériques standards, et des fonctions composites définies dans les bases de référence CEC'2005 et CEC'2013. L'analyse de performance est appuyée par une étude statistique commune à ce genre de problèmes, ainsi qu'une étude de robustesse illustrant l'impact de certains paramètres de conception sur la performance de l'approche proposée. Les résultats obtenus affichent dans l'ensemble une nette supériorité des modèles d'optimisation développés, dont la performance revient en grande partie aux différents mécanismes comportementaux introduits, et qui ont positivement impacté les capacités de diversification et d'intensification du processus itératif de recherche globale de la stratégie développée.

Une étude de validation du modèle mCLABC (modified CLABC) proposé a été ensuite entamée par la considération de six problèmes d'application relevant du domaine des systèmes intelligents et de l'ingénierie des systèmes. Les applications appréhendées concernent, tout précisément, l'identification des systèmes neuro-flous, la conception des classifieurs neuronaux, la classification de données non supervisée, la conception mécanique sous contraintes, la compensation de la non-linéarité d'un transducteur inductif LVDT et l'optimisation du contrôle PID. Le traitement de certaines de ces problèmes s'appuie sur des données expérimentales réelles.

Au vu de l'ensemble des résultats illustrés le long de cette thèse, il en ressort que la stratégie d'optimisation proposée s'est montrée imposante dans la plupart des cas d'application étudiés, ce qui conforte par excellence la contribution de la présente thèse. La faisabilité du modèle mCLABC en tant que nouvel outil d'optimisation est alors clairement démontrée**.** Nous pourrions ainsi affirmer avoir pu avancer dans le cadre de ce travail de recherche une approche originale du domaine de l'optimisation par essaims d'abeilles qui se rajoute confortablement à une variété de méthodes métaheuristiques existantes.

Comme perspectives de recherche, plusieurs pistes méritent d'être investies :

- L'adaptation de l'approche proposée pour une éventuelle implémentation sur plateformes de calcul parallèles. Ceci est d'une importance primordiale pour le calcul haute performance (High performance computing) qui aura pour effet d'accélérer le processus de convergence et de réduire le temps de calcul, notamment pour les problèmes de grande dimension.
- L'élaboration d'une version mCLABC adaptée aux problèmes d'optimisation multi-objectifs qui serait d'un intérêt majeur pour les problèmes de conception mécanique et de commande des procédés, pour ne citer que ces domaines.

- L'hybridation du modèle mCLABC avec d'autres approches d'optimisation de recherche globale ou de recherche locale, partant de certaines conclusions issues des analyses de performance et de robustesse illustrées dans ce travail de thèse.
- L'intégration de nouveaux concepts comportementaux propres aux essaims d'abeilles en vue d'une caractérisation meilleure du processus de fourragement collectif. A ce titre, il serait envisageable de recourir à une altération mixte des caractéristiques des sources de nourriture à opérer à travers la manipulation croisée des paramètres à optimiser. Ceci traduirait un effet de diversification assez marqué.
- Le développement d'une version mCLABC plutôt généralisée dédiée aux problèmes d'optimisation avec contraintes (tous types confondus).
- L'adaptation de l'approche d'optimisation par essaims d'abeilles à apprentissage coopératif à l'optimisation à variables discrètes.

Bibliographie

[1] V. Paschos, “Optimisation combinatoire. 1, Concepts fondamentaux,” *Hermès Sci. Lavoisier Paris*, 2005.

[2] E.-G. Talbi, *Metaheuristics: from design to implementation*, vol. 74. John Wiley & Sons, 2009.

[3] P. Siarry, *Metaheuristics*. Springer, 2016.

[4] J. Dréo, “Adaptation de la méthode des colonies de fourmis pour l’optimisation en variables continues. Application en génie biomédical.” PhD thesis, Paris 12, 2004.

[5] C. H. Papadimitriou, “Computational complexity. 1994.” Addison-Wesley.

[6] D. L. Applegate, W. J. Cook, S. Dash, and D. S. Johnson, “A Practical Guide to Discrete Optimization.” Print, 2014.

[7] J. Dréo, A. Pétrowski, P. Siarry, and E. Taillard, *Metaheuristics for hard optimization: methods and case studies*. Springer Science & Business Media, 2006.

[8] G. Polya, “How to solve it: A new aspect of mathematical model,” *Princeton, New Jersey*, 1945.

[9] M. Mitchell, J. H. Holland, and S. Forrest, “When will a genetic algorithm outperform hill climbing,” in *Advances in neural information processing systems*, 1994, pp. 51–58.

[10] J.-C. Boisson, “Modélisation et résolution par métaheuristiques coopératives: de l’atome à la séquence protéique.” Université Lille 1, 2008.

[11] A. S. Fraser, “Simulation of genetic systems by automatic digital computers I. Introduction,” *Aust. J. Biol. Sci.*, vol. 10, no. 4, pp. 484–491, 1957.

[12] J. H. Holland, “Adaptation in natural and artificial systems. An introductory analysis with application to biology, control, and artificial intelligence,” *Ann Arbor, MI Univ. Michigan Press*, pp. 439–444, 1975.

[13] R. Storn and K. Price, “Differential Evolution-A simple and efficient adaptive scheme for global optimization over continuous spaces,” 1995.

[14] K. Price, R. M. Storn, and J. A. Lampinen, “Differential Evolution: A Practical Approach to Global Optimization (Natural Computing Series),” 2005.

[15] E. Rechenberg, “Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. 1973,” *Frommann-Holzboog Verlag, Stuttgart*.

[16] L. J. Fogel, A. J. Owens, and M. J. Walsh, “Artificial intelligence through simulated evolution,” 1966.

[17] K.-F. Man, K. S. TANG, and S. Kwong, *Genetic Algorithms: Concepts and Designs*. Springer Science & Business Media, 2001.

[18] S. N. Sivanandam and S. N. Deepa, *Introduction to genetic algorithms*. Springer Science & Business Media, 2007.

[19] E. Bonabeau, D. de R. D. F. Marco, M. Dorigo, G. Théraulaz, and G. Theraulaz, *Swarm intelligence: from natural to artificial systems*, no. 1. Oxford university press, 1999.

[20] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: optimization by a colony of cooperating agents," *IEEE Trans. Syst. Man, Cybern. Part B*, vol. 26, no. 1, pp. 29–41, 1996.

[21] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Micro Machine and Human Science, 1995. MHS'95., Proceedings of the Sixth International Symposium on*, 1995, pp. 39–43.

[22] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," Technical report-tr06, Erciyes university, engineering faculty, computer engineering department, 2005.

[23] X.-S. Yang, "A new metaheuristic bat-inspired algorithm," in *Nature inspired cooperative strategies for optimization (NICSO 2010)*, Springer, 2010, pp. 65–74.

[24] S. Goss, S. Aron, J.-L. Deneubourg, and J. M. Pasteels, "Self-organized shortcuts in the Argentine ant," *Naturwissenschaften*, vol. 76, no. 12, pp. 579–581, 1989.

[25] M. R. Bonyadi and Z. Michalewicz, "Particle swarm optimization for single objective continuous space problems: a review." MIT Press, 2017.

[26] H. A. Abbass, "MBO: Marriage in honey bees optimization-A haplometrosis polygynous swarming approach," in *Proceedings of the 2001 Congress on Evolutionary Computation CEC2001*, 2001, pp. 207–214.

[27] O. B. Haddad, A. Afshar, and M. A. Mariño, "Honey-bees mating optimization (HBMO) algorithm: a new heuristic approach for water resources optimization," *water Resour. Manag.*, vol. 20, no. 5, pp. 661–680, 2006.

[28] C. Yang, J. Chen, and X. Tu, "Algorithm of fast marriage in honey bees optimization and convergence analysis," in *Automation and Logistics, 2007 IEEE International Conference on*, 2007, pp. 1794–1799.

[29] P. Ćurković and B. Jerbić, "Honey-bees optimization algorithm applied to path planning problem," *Int. J. Simul. Model.*, vol. 6, no. 3, p. 154, 2007.

[30] T. Sato and M. Hagiwara, "Bee system: finding solution by a concentrated search," *IEEJ Trans. Electron. Inf. Syst.*, vol. 118, no. 5, pp. 721–726, 1998.

[31] D. Teodorovic and M. Dell'Orco, "Bee colony optimization–a cooperative learning approach to complex transportation problems," *Adv. OR AI methods Transp.*, vol. 51, p. 60, 2005.

[32] D. Karaboga and B. Basturk, "On the performance of artificial bee colony (ABC) algorithm," *Appl. Soft Comput.*, vol. 8, no. 1, pp. 687–697, 2008.

[33] D. T. Pham, A. Ghanbarzadeh, E. Koc, S. Otri, S. Rahim, and M. Zaidi, "The bees

algorithm. Technical note," *Manuf. Eng. Centre, Cardiff Univ. UK*, pp. 1–57, 2005.

[34] B. Yuce, M. S. Packianather, E. Mastrocinque, D. T. Pham, and A. Lambiase, "Honey bees inspired optimization method: the bees algorithm," *Insects*, vol. 4, no. 4, pp. 646–662, 2013.

[35] A. Gogna and A. Tayal, "Metaheuristics: review and application," *J. Exp. Theor. Artif. Intell.*, vol. 25, no. 4, pp. 503–526, 2013.

[36] X.-S. Yang and M. Karamanoglu, "Swarm intelligence and bio-inspired computation: an overview," in *Swarm Intelligence and Bio-Inspired Computation*, Elsevier, 2013, pp. 3–23.

[37] X.-S. Yang and A. Hossein Gandomi, "Bat algorithm: a novel approach for global engineering optimization," *Eng. Comput.*, vol. 29, no. 5, pp. 464–483, 2012.

[38] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer, "Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems," *IEEE Trans. Evol. Comput.*, vol. 10, no. 6, pp. 646–657, 2006.

[39] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Trans. Evol. Comput.*, vol. 13, no. 2, pp. 398–417, 2009.

[40] M. A. Contreras-Cruz, V. Ayala-Ramirez, and U. H. Hernandez-Belmonte, "Mobile robot path planning using artificial bee colony and evolutionary programming," *Appl. Soft Comput.*, vol. 30, pp. 319–328, 2015.

[41] D. Karaboga and B. Akay, "Proportional—integral—derivative controller design by using artificial bee colony, harmony search, and the bees algorithms," *Proc. Inst. Mech. Eng. Part I J. Syst. Control Eng.*, vol. 224, no. 7, pp. 869–883, 2010.

[42] P. Dash, L. C. Saikia, and N. Sinha, "Automatic generation control of multi area thermal system using Bat algorithm optimized PD–PID cascade controller," *Int. J. Electr. Power Energy Syst.*, vol. 68, pp. 364–372, 2015.

[43] B. Bhanu, S. Lee, and J. Ming, "Adaptive image segmentation using a genetic algorithm," *IEEE Trans. Syst. Man, Cybern.*, vol. 25, no. 12, pp. 1543–1567, 1995.

[44] J. R. Koza, F. H. Bennett, D. Andre, and M. A. Keane, "Automated design of both the topology and sizing of analog electrical circuits using genetic programming," in *Artificial Intelligence in Design'96*, Springer, 1996, pp. 151–170.

[45] T. J. Ai and V. Kachitvichyanukul, "A particle swarm optimization for the vehicle routing problem with simultaneous pickup and delivery," *Comput. Oper. Res.*, vol. 36, no. 5, pp. 1693–1702, 2009.

[46] S. P. Mendes, J. A. G. Pulido, M. A. V. Rodriguez, M. D. J. Simon, and J. M. S. Perez, "A differential evolution based algorithm to optimize the radio network design problem," in *e-Science and Grid Computing, 2006. e-Science'06. Second IEEE International Conference on*, 2006, p. 119.

[47] M. Tuba, A. Alihodzic, and N. Bacanin, "Cuckoo search and bat algorithm applied to training feed-forward neural networks," in *Recent Advances in Swarm Intelligence and*

Evolutionary Computation, Springer, 2015, pp. 139–162.

[48] H. Habbi, Y. Boudouaoui, D. Karaboga, and C. Ozturk, "Self-generated fuzzy systems design using artificial bee colony optimization," *Inf. Sci. (Ny).*, vol. 295, pp. 145–159, 2015.

[49] L. M. Abualigah, A. T. Khader, E. S. Hanandeh, and A. H. Gandomi, "A novel hybridization strategy for krill herd algorithm applied to clustering techniques," *Appl. Soft Comput.*, vol. 60, pp. 423–435, 2017.

[50] A. Nasrollahzadeh, G. Karimian, and A. Mehrafsa, "Implementation of neuro-fuzzy system with modified high performance genetic algorithm on embedded systems," *Appl. Soft Comput.*, vol. 60, pp. 602–612, 2017.

[51] B. K. Panigrahi, Y. Shi, and M.-H. Lim, *Handbook of swarm intelligence: concepts, principles and applications*, vol. 8. Springer Science & Business Media, 2011.

[52] P. Siarry, *Metaheuristics*. Springer, 2017.

[53] D. Karaboga and B. Akay, "A comparative study of artificial bee colony algorithm," *Appl. Math. Comput.*, vol. 214, no. 1, pp. 108–132, 2009.

[54] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm," *J. Glob. Optim.*, vol. 39, no. 3, pp. 459–471, Oct. 2007.

[55] G. Zhu and S. Kwong, "Gbest-guided artificial bee colony algorithm for numerical function optimization," *Appl. Math. Comput.*, vol. 217, no. 7, pp. 3166–3173, 2010.

[56] A. Banharnsakun, T. Achalakul, and B. Sirinaovakul, "The best-so-far selection in artificial bee colony algorithm," *Appl. Soft Comput.*, vol. 11, no. 2, pp. 2888–2901, 2011.

[57] S. Kumar, V. K. Sharma, and R. Kumari, "A novel hybrid crossover based artificial bee colony algorithm for optimization problem," *arXiv Prepr. arXiv1407.5574*, 2014.

[58] H. Duan, C. Xu, and Z.-H. Xing, "A hybrid artificial bee colony optimization and quantum evolutionary algorithm for continuous optimization problems," *Int. J. Neural Syst.*, vol. 20, no. 01, pp. 39–50, 2010.

[59] B. Akay and D. Karaboga, "A modified artificial bee colony algorithm for real-parameter optimization," *Inf. Sci. (Ny).*, vol. 192, pp. 120–142, 2012.

[60] Q.-K. Pan, M. F. Tasgetiren, P. N. Suganthan, and T. J. Chua, "A discrete artificial bee colony algorithm for the lot-streaming flow shop scheduling problem," *Inf. Sci. (Ny).*, vol. 181, no. 12, pp. 2455–2468, 2011.

[61] D. Karaboga and B. Basturk, "Artificial bee colony (ABC) optimization algorithm for solving constrained optimization problems," in *International fuzzy systems association world congress*, 2007, pp. 789–798.

[62] W. Zou, Y. Zhu, H. Chen, and B. Zhang, "Solving multiobjective optimization problems using artificial bee colony algorithm," *Discret. Dyn. Nat. Soc.*, vol. 2011, 2011.

[63] H. Narasimhan, "Parallel artificial bee colony (PABC) algorithm," in *Nature & Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on*, 2009, pp. 306–311.

[64] D. Karaboga, B. Akay, and C. Ozturk, "Artificial bee colony (ABC) optimization algorithm for training feed-forward neural networks," in *International conference on modeling decisions for artificial intelligence*, 2007, pp. 318–329.

[65] B. Akay and D. Karaboga, "Artificial bee colony algorithm for large-scale problems and engineering design optimization," *J. Intell. Manuf.*, vol. 23, no. 4, pp. 1001–1014, 2012.

[66] K. Balasubramani and K. Marcus, "Artificial bee colony algorithm to improve brain MR image segmentation," *Int. J. Comput. Sci. Eng.*, vol. 5, no. 1, p. 31, 2013.

[67] H. Gozde, M. C. Taplamacioglu, and I. Kocaarslan, "Application of artificial bee colony algorithm in an automatic voltage regulator (AVR) system," *Int. J. Tech. Phys. Probl. Eng.*, vol. 1, no. 3, pp. 88–92, 2010.

[68] M. Yin, X. Li, and J. Zhou, "An efficient job shop scheduling algorithm based on artificial bee colony," *Sci. Res. Essays*, vol. 6, no. 12, pp. 2578–2596, 2011.

[69] B. S. Prajapati and L. Srivastava, "Multi-Objective reactive power optimization using artificial bee colony algorithm," *Int. J. Eng. Innov. Technol. pp*, pp. 126–131, 2012.

[70] A. L. BOLAJI, A. T. Khader, M. A. Al-Betar, and M. A. Awadallah, "Artificial bee colony algorithm, its variants and applications: A survey.," *J. Theor. Appl. Inf. Technol.*, vol. 47, no. 2, 2013.

[71] D. Karaboga, B. Gorkemli, C. Ozturk, and N. Karaboga, "A comprehensive survey: artificial bee colony (ABC) algorithm and applications," *Artif. Intell. Rev.*, vol. 42, no. 1, pp. 21–57, 2014.

[72] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 67–82, 1997.

[73] V. Gardeux, "Conception d'heuristiques d'optimisation pour les problèmes de grande dimension: application à l'analyse de données de puces à ADN." Université Paris-Est, 2011.

[74] M. Jamil and X.-S. Yang, "A literature survey of benchmark functions for global optimisation problems," *Int. J. Math. Model. Numer. Optim.*, vol. 4, no. 2, pp. 150–194, 2013.

[75] Y. Cooren, "Perfectionnement d'un algorithme adaptatif d'optimisation par essaim particulaire: application en génie médical et en électronique." Université Paris-Est, 2008.

[76] P. N. Suganthan *et al.*, "Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization," *KanGAL Rep.*, vol. 2005005, p. 2005, 2005.

[77] J. J. Liang, B. Y. Qu, P. N. Suganthan, and A. G. Hernández-Díaz, "Problem definitions and evaluation criteria for the CEC 2013 special session on real-parameter

optimization," *Comput. Intell. Lab. Zhengzhou Univ. Zhengzhou, China Nanyang Technol. Univ. Singapore, Tech. Rep.*, vol. 201212, pp. 3–18, 2013.

[78] W. Gao, S. Liu, and L. Huang, "A global best artificial bee colony algorithm for global optimization," *J. Comput. Appl. Math.*, vol. 236, no. 11, pp. 2741–2753, 2012.

[79] F. S. Abu-Mouti and M. E. El-Hawary, "Overview of Artificial Bee Colony (ABC) algorithm and its applications," in *Systems Conference (SysCon), 2012 IEEE International*, 2012, pp. 1–6.

[80] W. Gao, L. Huang, S. Liu, F. T. S. Chan, C. Dai, and X. Shan, "Artificial bee colony algorithm with multiple search strategies," *Appl. Math. Comput.*, vol. 271, pp. 269–287, 2015.

[81] M. S. Kiran, H. Hakli, M. Gunduz, and H. Uguz, "Artificial bee colony algorithm with variable search strategy for continuous optimization," *Inf. Sci. (Ny).*, vol. 300, pp. 140–157, 2015.

[82] F. Harfouchi, H. Habbi, C. Ozturk, and D. Karaboga, "Modified multiple search cooperative foraging strategy for improved artificial bee colony optimization with robustness analysis," *Soft Comput.*, vol. 22, pp. 6371–6394, 2018.

[83] F. Harfouchi and H. Habbi, "A cooperative learning artificial bee colony algorithm with multiple search mechanisms," *Int. J. Hybrid Intell. Syst.*, vol. 13, no. 2, pp. 113–124, 2016.

[84] W. Gao, S. Liu, and L. Huang, "A novel artificial bee colony algorithm based on modified search equation and orthogonal learning," *IEEE Trans. Cybern.*, vol. 43, no. 3, pp. 1011–1024, 2013.

[85] C. Ozturk, E. Hancer, and D. Karaboga, "A novel binary artificial bee colony algorithm based on genetic operators," *Inf. Sci. (Ny).*, vol. 297, pp. 154–170, 2015.

[86] C. Y. Chung, H. Yu, and K. P. Wong, "An advanced quantum-inspired evolutionary algorithm for unit commitment," *IEEE Trans. Power Syst.*, vol. 26, no. 2, pp. 847–854, 2011.

[87] M. H. Kashan, N. Nahavandi, and A. H. Kashan, "DisABC: a new artificial bee colony algorithm for binary optimization," *Appl. Soft Comput.*, vol. 12, no. 1, pp. 342–352, 2012.

[88] Y. Wang, H.-X. Li, T. Huang, and L. Li, "Differential evolution based on covariance matrix learning and bimodal distribution parameter setting," *Appl. Soft Comput.*, vol. 18, pp. 232–247, 2014.

[89] Y. Wang, Z. Cai, and Q. Zhang, "Differential evolution with composite trial vector generation strategies and control parameters," *IEEE Trans. Evol. Comput.*, vol. 15, no. 1, pp. 55–66, 2011.

[90] W.-F. Gao, S.-Y. Liu, and L.-L. Huang, "A novel artificial bee colony algorithm with Powell's method," *Appl. Soft Comput.*, vol. 13, no. 9, pp. 3763–3775, 2013.

[91] T.-K. Dao, S.-C. Chu, C.-S. Shieh, and M.-F. Horng, "Compact artificial bee colony," in *International Conference on Industrial, Engineering and Other Applications of*

Applied Intelligent Systems, 2014, pp. 96–105.

[92] A. Banitalebi, M. I. A. Aziz, A. Bahar, and Z. A. Aziz, "Enhanced compact artificial bee colony," *Inf. Sci. (Ny).*, vol. 298, pp. 491–511, 2015.

[93] F. Neri, "Memetic compact differential evolution for cartesian robot control," *IEEE Comput. Intell. Mag.*, vol. 5, no. 2, pp. 54–65, 2010.

[94] B. Haznedar and A. Kalinli, "Training ANFIS using genetic algorithm for dynamic systems identification," *Int. J. Intell. Syst. Appl. Eng.*, vol. 4, no. Special Issue-1, pp. 44–47, 2016.

[95] D. Karaboga and E. Kaya, "An adaptive and hybrid artificial bee colony algorithm (aABC) for ANFIS training," *Appl. Soft Comput.*, vol. 49, pp. 423–436, 2016.

[96] J.-S. Jang, "ANFIS: adaptive-network-based fuzzy inference system," *IEEE Trans. Syst. Man. Cybern.*, vol. 23, no. 3, pp. 665–685, 1993.

[97] G. P. Zhang, "Neural networks for classification: a survey," *IEEE Trans. Syst. Man, Cybern. Part C (Applications Rev.*, vol. 30, no. 4, pp. 451–462, 2000.

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

[99] S. Haykin, *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1994.

[100] S. Dominique, "Algorithme intelligent d'optimisation d'un design structurel de grande envergure." École Polytechnique de Montréal, 2009.

[101] P. Comon, "Classification supervisée par réseaux multicouches," *Trait. du signal*, vol. 8, no. 6, pp. 387–407, 1991.

[102] R. Mendes, P. Cortez, M. Rocha, and J. Neves, "Particle swarms for feedforward neural network training," in *Neural Networks, 2002. IJCNN'02. Proceedings of the 2002 International Joint Conference on*, 2002, vol. 2, pp. 1895–1899.

[103] Z.-G. Che, T.-A. Chiang, and Z.-H. Che, "Feed-forward neural networks training: a comparison between genetic algorithm and back-propagation learning algorithm," *Int. J. Innov. Comput. Inf. Control*, vol. 7, no. 10, pp. 5839–5850, 2011.

[104] J. Ilonen, J.-K. Kamarainen, and J. Lampinen, "Differential evolution training algorithm for feed-forward neural networks," *Neural Process. Lett.*, vol. 17, no. 1, pp. 93–105, 2003.

[105] D. Karaboga and C. Ozturk, "Neural networks training by artificial bee colony algorithm on pattern classification," *Neural Netw. World*, vol. 19, no. 3, p. 279, 2009.

[106] S. E. Fahlman, "An empirical study of learning speed in back-propagation networks," 1988.

[107] D. Dheeru and E. K. Taniskidou, "UCI machine learning repository (2017)," *URL http//archive. ics. uci. edu/ml*, 2017.

[108] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, p. 533, 1986.

[109] M. T. Hagan and M. B. Menhaj, “Training feedforward networks with the Marquardt algorithm,” *IEEE Trans. Neural Networks*, vol. 5, no. 6, pp. 989–993, 1994.

[110] R. Storn and K. Price, “Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces,” *J. Glob. Optim.*, vol. 11, no. 4, pp. 341–359, 1997.

[111] J. Kenney, “Particle swarm optimization,” in *Proc. IEEE International Conference on Neural Networks, 1995*, 1995, pp. 1942–1948.

[112] J. A. Bullinaria and K. AlYahya, “Artificial bee colony training of neural networks,” in *Nature Inspired Cooperative Strategies for Optimization (NICSO 2013)*, Springer, 2014, pp. 191–201.

[113] D. Karaboga and C. Ozturk, “A novel clustering approach: Artificial Bee Colony (ABC) algorithm,” *Appl. Soft Comput.*, vol. 11, no. 1, pp. 652–657, 2011.

[114] A. Kumar, D. Kumar, and S. K. Jarial, “A Review on Artificial Bee Colony Algorithms and Their Applications to Data Clustering,” *Cybern. Inf. Technol.*, vol. 17, no. 3, pp. 3–28, 2017.

[115] P. S. Shelokar, V. K. Jayaraman, and B. D. Kulkarni, “An ant colony approach for clustering,” *Anal. Chim. Acta*, vol. 509, no. 2, pp. 187–195, 2004.

[116] O. Moh’d Alia, M. A. Al-Betar, R. Mandava, and A. T. Khader, “Data clustering using harmony search algorithm,” in *International Conference on Swarm, Evolutionary, and Memetic Computing*, 2011, pp. 79–88.

[117] S. Rana, S. Jasola, and R. Kumar, “A review on particle swarm optimization algorithms and their applications to data clustering,” *Artif. Intell. Rev.*, vol. 35, no. 3, pp. 211–222, 2011.

[118] R. Jensi and G. W. Jiji, “An improved krill herd algorithm with global exploration capability for solving numerical function optimization problems and its application to data clustering,” *Appl. Soft Comput.*, vol. 46, pp. 230–245, 2016.

[119] K. Deb, “An efficient constraint handling method for genetic algorithms,” *Comput. Methods Appl. Mech. Eng.*, vol. 186, no. 2–4, pp. 311–338, 2000.

[120] X. Hu and R. Eberhart, “Solving constrained nonlinear optimization problems with particle swarm optimization,” in *Proceedings of the sixth world multiconference on systemics, cybernetics and informatics*, 2002, vol. 5, pp. 203–206.

[121] C. A. C. Coello, “Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art,” *Comput. Methods Appl. Mech. Eng.*, vol. 191, no. 11–12, pp. 1245–1287, 2002.

[122] D. Karaboga and B. Akay, “A modified artificial bee colony (ABC) algorithm for constrained optimization problems,” *Appl. Soft Comput.*, vol. 11, no. 3, pp. 3021–3031, 2011.

[123] I. Brajevic and M. Tuba, “An upgraded artificial bee colony (ABC) algorithm for constrained optimization problems,” *J. Intell. Manuf.*, vol. 24, no. 4, pp. 729–740, 2013.

[124] S. Babaeizadeh and R. Ahmad, "An efficient artificial bee colony algorithm for constrained optimization problems," *J. Eng. Appl. Sci.*, vol. 9, no. 10–12, pp. 405–413, 2014.

[125] T. Ray and K. M. Liew, "Society and civilization: An optimization algorithm based on the simulation of social behavior," *IEEE Trans. Evol. Comput.*, vol. 7, no. 4, pp. 386–396, 2003.

[126] S. He, E. Prempain, and Q. H. Wu, "An improved particle swarm optimizer for mechanical design optimization problems," *Eng. Optim.*, vol. 36, no. 5, pp. 585–605, 2004.

[127] E. Mezura-Montes and C. A. C. Coello, "Useful infeasible solutions in engineering optimization with evolutionary algorithms," in *Mexican International Conference on Artificial Intelligence*, 2005, pp. 652–662.

[128] K. E. Parsopoulos and M. N. Vrahatis, "Unified particle swarm optimization for solving constrained engineering optimization problems," in *International conference on natural computation*, 2005, pp. 582–591.

[129] S. K. Mishra, G. Panda, and D. P. Das, "A novel method of extending the linearity range of linear variable differential transformer using artificial neural network," *IEEE Trans. Instrum. Meas.*, vol. 59, no. 4, pp. 947–953, 2010.

[130] P. Veeraian, U. Gandhi, and U. Mangalanathan, "Fractional order linear variable differential transformer: design and analysis," *AEU-International J. Electron. Commun.*, vol. 79, pp. 141–150, 2017.

[131] S. Murugan and S. P. Umayal, "An Evolutionary Optimized Algorithm Approach to Compensate the Non-linearity in Linear Variable Displacement Transducer Characteristics," *J. Electr. Eng. Technol.*, vol. 9, no. 6, pp. 2142–2153, 2014.

[132] S. Das, D. P. Das, and S. K. Behera, "Enhancing the linearity of LVDT by two-stage functional link artificial neural network with high accuracy and precision," in *Industrial Electronics and Applications (ICIEA), 2013 8th IEEE Conference on*, 2013, pp. 1358–1363.

[133] P. Misra, S. K. Mohini, and S. K. Mishra, "The Design and Implementation of an ANN-based Non-linearity Compensator of LVDT Sensor," *arXiv Prepr. arXiv1407.0506*, 2014.

[134] J. C. Patra, R. N. Pal, R. Baliarsingh, and G. Panda, "Nonlinear channel equalization for QAM signal constellation using artificial neural networks," *IEEE Trans. Syst. Man, Cybern. Part B*, vol. 29, no. 2, pp. 262–271, 1999.

[135] K. J. Åström and T. Hägglund, "Benchmark systems for PID control," *IFAC Proc. Vol.*, vol. 33, no. 4, pp. 165–166, 2000.

Annexe A

Les fonctions benchmarks standards

Fonction		Espace de recherche	Type	Formule mathématique	Seuil
F1	Sphere	$[-100,100]^D$	US	$f_1(x)=\sum_{i=1}^{D}(x_i)^2$	1×10^{-8}
F2	Elliptic	$[-100,100]^D$	UN	$f_2(x)=\sum_{i=1}^{D}(10^6)^{(i-1)/(n-1)}x_i^2$	1×10^{-8}
F3	SumSquares	$[-10,10]^D$	US	$f_3(x)=\sum_{i=1}^{D}ix_i^2$	1×10^{-8}
F4	SumPower	$[-10,10]^D$	MS	$f_4(x)=\sum_{i=1}^{D}\lvert x_i\rvert^{(i+1)}$	1×10^{-8}
F5	Schwefel2.22	$[-10,10]^D$	UN	$f_5(x)=\sum_{i=1}^{D}\lvert x_i\rvert+\prod_{i=1}^{D}\lvert x_i\rvert$	1×10^{-8}
F6	Schwefel2.21	$[-100,100]^D$	UN	$f_6(x)=\max_i\{\lvert x_i\rvert,1\le i\le n\}$	4×10^{-1}
F7	Step	$[-100,100]^D$	US	$f_7(x)=\sum_{i=1}^{D}\left(\lfloor x_i+0.5\rfloor\right)^2$	1×10^{-8}
F8	Quartic	$[-1.28,1.28]^D$	US	$f_8(x)=\sum_{i=1}^{D}ix_i^4$	1×10^{-8}
F9	QuarticWN	$[-1.28,1.28]^D$	US	$f_9(x)=\sum_{i=1}^{D}ix_i^4+random[0,1)$	1×10^{-1}
F10	Rosenbrock	$[-30,30]^D$	UN	$f_{10}(x)=\sum_{i=1}^{D-1}[100(x_{i+1}-x_i^2)^2+(x_i-1)^2]$	5×10^{0}
F11	Rastrigin	$[-5.12,5.12]^D$	MS	$f_{11}(x)=\sum_{i=1}^{D}[x_i^2-10\cos(2\pi x_i)+10]$	1×10^{-8}
F12	Non-continuous rastrigin	$[-5.12,5.12]^D$	MS	$f_{12}(x)=\sum_{i=1}^{D}[y_i^2-10\cos(2\pi y_i)+10]$ $\begin{cases} x_i & \lvert x_i\rvert<\frac{1}{2} \\ \frac{round(2x_i)}{2} & \lvert x_i\rvert\ge\frac{1}{2} \end{cases}$	1×10^{-8}
F13	Griewank	$[-600,600]^D$	MN	$f_{13}(x)=\frac{1}{4000}\sum_{i=1}^{D}x_i^2-\prod_{i=1}^{D}\cos(\frac{x_i}{\sqrt{i}})+1$	1×10^{-8}
F14	Schaffer	$[-100,100]^D$	MN	$f_{14}(x)=0.5+\frac{\sin^2(\sqrt{x_1^2+x_2^2})-0.5}{(1+0.001(x_1^2+x_2^2))^2}$	1×10^{-1}
F15	Ackley	$[-32,32]^D$	MN	$f_{15}(x)=-20\exp\left\{-0.2\sqrt{\frac{1}{D}\sum_{i-1}^{D}x_i^2}\right\}-\exp\left\{\frac{1}{D}\sum_{i-1}^{D}\cos(2\pi x_i)\right\}+20+e$	1×10^{-8}
F16	Penalized1	$[-50,50]^D$	MN	$f_{16}(x)=\frac{\pi}{D}\left\{10\sin^2(\pi y_1)+\sum_{i=1}^{D-1}(y_i-1)^2\left[1+10\sin^2(\pi y_{i+1})\right]+(y_D-1)^2\right\}$ $+\sum_{i=1}^{D}u(x_i,10,100,4)$ $y_i=1+\frac{1}{4}(x_i+1)$ $u_{x_i,a.k.m}=\begin{cases} k(x_i-a)^m & x_i>a \\ 0 & -a\le x_i\le a \\ k(x_i-a)^m & x_i<-a \end{cases}$	1×10^{-8}
F17	Penalized2	$[-50,50]^D$	MN	$f_{17}(x)=\frac{1}{10}\left\{\sin^2(\pi x_1)+\sum_{i=1}^{D-1}(x_i-1)^2\left[1+\sin^2(3\pi x_{i+1})\right]+(x_D-1)^2[1+\sin^2(2\pi x_{i+1})]\right\}$ $+\sum_{i=1}^{D}u(x_i,5,100,4)$	1×10^{-8}
F18	Alpine	$[-10,10]^D$	MS	$f_{18}(x)=\sum_{i=1}^{D}\lvert x_i.\sin(x_i)+0.1.x_i\rvert$	1×10^{-8}
F19	Levy	$[-10,10]^D$	MN	$f_{19}(x)=\sum_{i=1}^{D-1}(x_i-1)^2\left[1+\sin^2(3\pi x_{i+1})\right]+\sin^2(3\pi x_1)+\lvert x_D-1\rvert\left[1+\sin^2(3\pi x_D)\right]$	1×10^{-8}
F20	Weierstrass	$[-0.5,0.5]^D$	MN	$f_{20}(x)=\sum_{i=1}^{D}\left(\sum_{k=0}^{k_{\max}}\left[a^k\cos\left(2\pi b^k(x_i+0.5)\right)\right]\right)-D\sum_{k=0}^{k_{\max}}\left[a^k\cos\left(2\pi b^k 0.5\right)\right]$, $a=0.5, b=3, k_{\max}=20$	1×10^{-8}
F21	Himmelblau	$[-5,5]^D$	MS	$f_{21}(x)=\frac{1}{D}\sum_{i=1}^{D}\left(x_i^4-16x_i^2+5x_i\right)$	-78
F22	Michalewicz	$[0,\pi]^D$	MS	$f_{22}(x)=-\sum_{i=1}^{D}\sin(x_i)\sin^{20}\left(\frac{i*x_i^2}{\pi}\right)$	-49, -95, -190

L'optimum global : $x^*=(0,0,..,0)$, $\quad f(x^*)=0$

U: fonction unimodale, M: fonction multimodale, S: fonction séparable, N: fonction non-séparable, Seuil : défini pour la fonction objectif utilisée pour les analyses statistiques.

Les fonctions composites CEC'05 [76]

Fonction		Désignation	Espace de recherche	$f(x^*)$
Unimodale	f1	Shifted Sphere Function	$[-100,100]^D$	-450
	f2	Shifted Schwefel's Problem 1.2	$[-100,100]^D$	-450
	f3	Shifted Rotated High Conditioned Elliptic Function	$[-100,100]^D$	-450
	f4	Shifted Schwefel's Problem 1.2 with Noise in Fitness	$[-100,100]^D$	-450
	f5	Schwefel's Problem 2.6 with Global Optimum on Bounds	$[-100,100]^D$	-310
Multimodale	f6	Shifted Rosenbrock's Function	$[-100,100]^D$	390
	f7	Shifted Rotated Griewank's Function without Bounds	$[-0,600]^D$	-180
	f8	Shifted Rotated Ackley's Function with Global Optimum on Bounds	$[-32,32]^D$	-140
	f9	Shifted Rastrigin's Function	$[-5,5]^D$	-330
	f10	Shifted Rotated Rastrigin's Function	$[-5,5]^D$	-330
	f11	Shifted Rotated Weierstrass Function	$[-0.5,0.5]^D$	90
	f12	Schwefel's Problem 2.13	$[-\pi,\pi]^D$	-460
Étendue	f13	Expanded Extended Griewank's plus Rosenbrock's Function (f8f2)	$[-3,1]^D$	-130
	f14	Shifted Rotated Expanded Scaffer's f6	$[-100,100]^D$	-300
Composite hybride	f15	Hybrid Composition Function	$[-5,5]^D$	120
	f16	Rotated Hybrid Composition Function	$[-5,5]^D$	120
	f17	Rotated Hybrid Composition Function with Noise in Fitness	$[-5,5]^D$	120
	f18	Rotated Hybrid Composition Function	$[-5,5]^D$	10
	f19	Rotated Hybrid Composition Function with a Narrow Basin for the Global Optimum	$[-5,5]^D$	10
	f20	Rotated Hybrid Composition Function with the Global Optimum on the Bounds	$[-5,5]^D$	10
	f21	Rotated Hybrid Composition Function	$[-5,5]^D$	360
	f22	Rotated Hybrid Composition Function with High Condition Number Matrix	$[-5,5]^D$	360
	f23	Non-Continuous Rotated Hybrid Composition Function	$[-5,5]^D$	360
	f24	Rotated Hybrid Composition Function	$[-5,5]^D$	260
	f25	Rotated Hybrid Composition Function without Bounds	$[-5,5]^D$	260

Les fonctions composites CEC'13 [77]

Fonction		**Désignation**	**$f(x^*)$**
Unimodale	f1	Sphere Function	-1400
	f2	Rotated High Conditioned Elliptic Function	-1300
	f3	Rotated Bent Cigar Function	-1200
	f4	Rotated Discus Function	-1100
	f5	Different Powers Function	-1000
Multimodale	f6	Rotated Rosenbrock's Function	-900
	f7	Rotated Schaffers f7 Function	-800
	f8	Rotated Ackley's Function	-700
	f9	Rotated Weierstrass Function	-600
	f10	Rotated Griewank's Function	-500
	f11	Rastrigin's Function	-400
	f12	Rotated Rastrigin's Function	-300
	f13	Non-Continuous Rotated Rastrigin's Function	-200
	f14	Schwefel's Function	-100
	f15	Rotated Schwefel's Function	100
	f16	Rotated Katsuura Function	200
	f17	Lunacek Bi_Rastrigin Function	300
	f18	Rotated Lunacek Bi_Rastrigin Function	400
	f19	Expanded Griewank's plus Rosenbrock's Function	500
	f20	Expanded Scaffer's f6 Function	600
Composite	f21	Composition Function 1 (n=5,Rotated)	700
	f22	Composition Function 2 (n=3,Unrotated)	800
	f23	Composition Function 3 (n=3,Rotated)	900
	f24	Composition Function 4 (n=3,Rotated)	1000
	f25	Composition Function 5 (n=3,Rotated)	1100
	f26	Composition Function 6 (n=5,Rotated)	1200
	f27	Composition Function 7 (n=5,Rotated)	1300
	f28	Composition Function 8 (n=5,Rotated)	1400
Espace de recherche : $[-100,100]^D$			

Annexe B

a) Observations expérimentales du LVDT-1

Déplacement du noyau (en mm)	Tension de sortie conditionnée (en mV)
-6.30	-2.91
-5.08	-2.38
-4.10	-1.87
-3.02	-1.35
-1.94	-0.84
-0.99	-0.34
0.05	0.15
1.20	0.90
1.20	0.90
2.21	1.90
3.20	3.00
4.22	4.10
5.92	2.67
6.37	3.20
7.46	3.72
8.49	4.21
9.51	4.70
10.57	5.21
11.53	5.69
12.68	6.22
13.73	6.72
14.70	7.22

b) Observations expérimentales du LVDT-2 (KL-68004) du laboratoire

Déplacement du noyau (en mm)	Tension de sortie conditionnée (en mV)
-5.18	-2.15
-4.93	-2.04
-4.44	-1.84
-3.51	-1.44
-3.03	-1.24
-1.72	-0.69
-1.19	-0.47
-0.92	-0.15
-0.13	-0.04
0.12	0.05
0.53	0.22
1.24	0.52
1.64	0.68
2.31	0.96
2.51	1.04
2.95	1.23
3.55	1.48
4.19	1.76
4.71	1.98
5.02	2.11
5.97	2.51
7.37	3.25