MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITE«ABBES LAGHROUR»DE KHENCHELA
FACULTE DES SCIENCES ET DE LA TECHNOLOGIE

Département de Génie industriel

N° de série :……

Mémoire de fin d'études

Pour l'obtention du diplôme de Master (L.M.D)

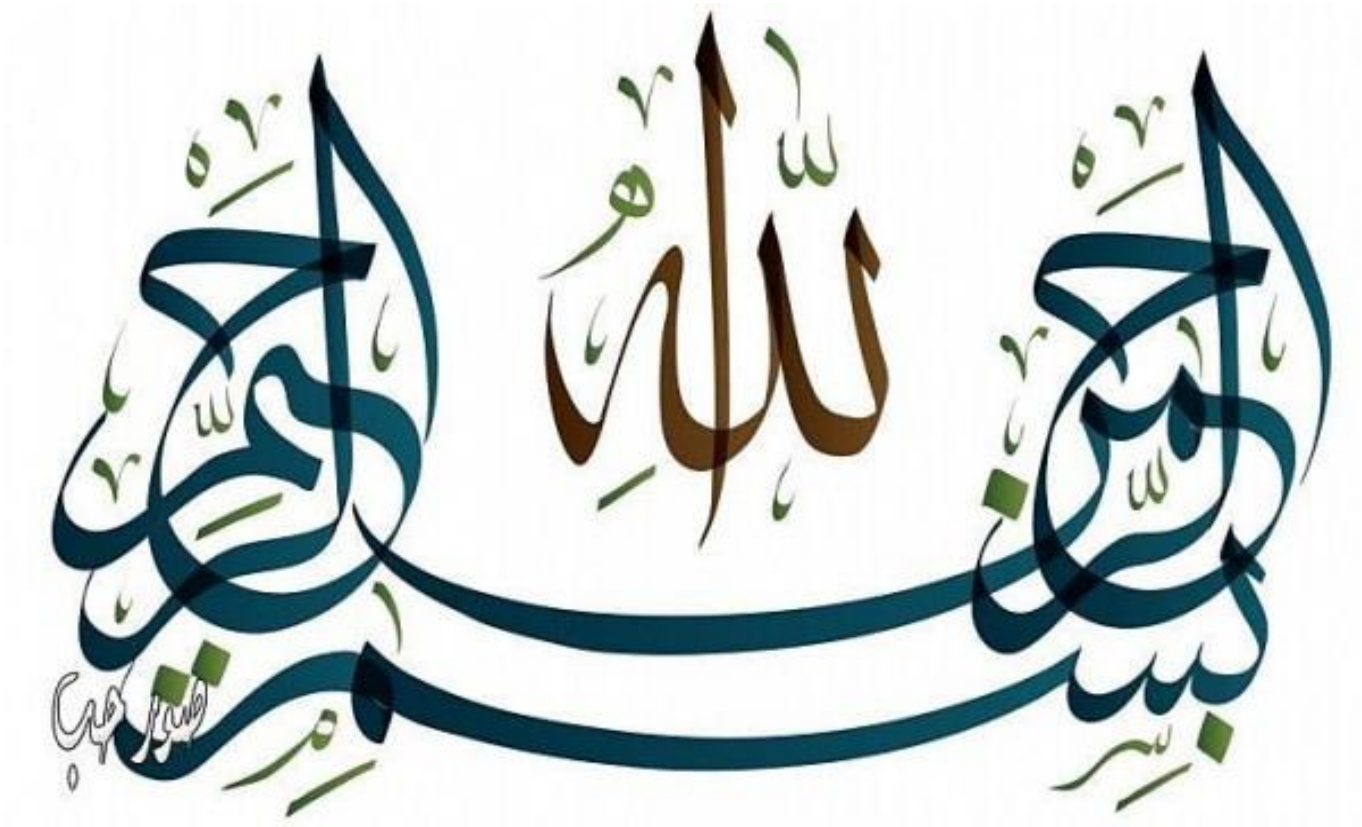
Filière : Automatique
Spécialité : Automatique et Informatique Industriel

Développement et application d'une nouvelle variante d'un algorithme de pollinisation des fleurs

Réalisé par : Bouzekri Kamel ***Dirigé par : Mr : Allouani Fouad***

Membres de jury :
Mr : S.Bououden ***MAA Président***
Mr : H.Berkani ***MCB Examinateur***
Mr : A.Menadi ***MAB Examinateur***

Présenté le : 29/06/2019


بسم الله الرحمن الرحيم

Remerciements

Je tiens à remercier avant tout, nôtre dieu pour m'avoir orienté vers le bon chemin et de m'avoir offert le climat et les conditions pour atteindre ce niveau d'étude et la réalisation de ce travail.

Je tiens à remercier aussi mes parents pour leur contribution, leur soutien et leur patience.

Je remercie mon encadreur, **Mr .ALLOUANI Fouad** pour ses conseils et ses soutiens tout le long de la préparation de ce mémoire.

Je remercie aussi l'ensemble de mes enseignants du département de Génie Industriel, pour les efforts qu'ils ont fournis durant mon cursus afin de m'amener jusqu'au bout de la formation.

J'adresse aussi mes remerciements à mon enseignent **Mr. BOURAS Mustafa**.

Je remercie aussi les membres de jury qui me font l'honneur de présider et d'examiner ce modeste travail.

Je profite aussi l'occasion pour remercier tous ceux qui ont collaboré de près ou de loin à la réalisation de ce travail.

Enfin, j'adresse mes plus sincères remerciements à tous mes proches et amis, pour leurs encouragements au cours de la réalisation de ce mémoire.

Dédicace

Je dédie ce travail à : Ma mère qui m'a entouré d'amour, d'affection et qui fait tout pour ma réussite, que dieu la garde,

Mon père qui m'a aidé à devenir ce que je suis aujourd'hui, que dieu le garde et le protège,

- ✓ Mes très chers frères et sœurs,
- ✓ Et à toute ma famille BOUZEKRI,
- ✓ Tous mes Amis,
- ✓ À tous mes amis de promotion master Automatique 2018 / 2019,
- ✓ A tous ceux que j'aime.

BOUZEKRI KAMEL.

Promotion 2019

Résumé

Dans ce mémoire, une variante de métaheuristique dite ; FPA-GA est proposée pour l'algorithme de pollinisation des fleurs (dit en anglais ; Flower Pollination Algorithme FPA). En effet, le but de cette contribution est de faire améliorer la version standard de ce dernier. L'idée principale du travail réalisé, est basée sur l'intégration en particulier de deux opérateurs d'algorithmes génétiques (*Genetic Algorithms* GAs) dits respectivement ; Opérateur de Croisement et celui de Mutation dans la boucle du programme principal de l'algorithme FPA.

L'algorithme développé est testé en utilisant un ensemble de fonctions de tests dites ; *fonctions benchmarks*. Les résultats de simulations, obtenus montrent l'efficacité de la méthode d'optimisation proposée.

Liste des Figures

Chapitre 1 : Généralités : métaheuristiques à population de solution

Chapitre 2 : Algorithme de pollinisation (Flower Pollination Algorithme-FPA)

Chapitre 3 : Les algorithmes génétiques (Genetic Algorithms)

Chapitre 4 : Développement d'une nouvelle variante de FPA et le FPA-GA

Chapitre 5 : Validation de la nouvelle variante de FPA ; le FPA-GA

Table des matières

CHAPITRE 3

Les algorithmes génétiques (Genetic Algorithms)

CHAPITRE 4

Développement d'une nouvelle variante de FPA ; le FPA-GA

CHAPITRE 5

Validation d'une nouvelle variante de FPA ; le FPA-GA

Introduction générale

Introduction générale

Il y'a long temps que, les problèmes d'optimisation prennent une place importante dans les préoccupations de chercheurs qui ont des intérêts à résoudre des difficultés résident dans le mode réel. En effet, avec le changement permanant dans le style de vie de l'être humain, par effet d'une recherche contenue d'une amélioration de cette dernière, un appel constant à ce domaine scientifique est toujours existé. Par le fait, cet appel est le motif principal de l'intérêt exceptionnel à ce genre d'étude.

Par définition, un problème d'optimisation est défini par un ensemble de variables, une fonction objectif et un ensemble de contraintes. Les variables du problème dites aussi variables de conception ou de décision peuvent être de différentes nature (réelle, entière, booléenne. etc.) et peuvent exprimer des données qualitatives ou quantitatives. L'espace d'état, appelé aussi domaine de recherche, représente l'enveloppe de toutes les solutions possibles du problème. Il est en général fini, puisque les méthodes de résolution opèrent dans des espaces bornés. Pour des raisons pratiques et de temps de calcul, cet espace doit être fini. Cette limitation n'est pas gênante, puisqu'en général le concepteur défini a priori le domaine de définition de chaque variable. Enfin, même dans le cas des problèmes à variables continues, une certaine granularité est définie. Ainsi, la fonction objectif définit le but à atteindre, on cherche à minimiser ou à maximiser celle dernière. L'ensemble des contraintes est en général un ensemble d'égalités ou d'inégalités que les variables de l'espace de recherche doivent satisfaire. Ces contraintes imposent généralement les limites de l'espace de recherche.

Au sens général, les méthodes d'optimisation travaillent à chercher un point ou un ensemble de points dans l'espace de recherche satisfaisant l'ensemble des contraintes, et maximisant ou minimisant la fonction objectif. Parmi ces méthodes, celles dites ; *métaheuristiques* qui sont des algorithmes généraux d'optimisation applicables à une grande variété de problèmes. Elles sont utilisées généralement quand les méthodes classiques ne sont plus efficaces. Les métaheuristiques ont un caractère stochastique, c.-à-d. qu'une partie de la recherche est conduite de façon aléatoire, elles sont inspirées d'analogies avec la réalité : physique (recuit simulé,...), biologie (algorithmes évolutionnaires, recherche tabou,...) ou éthologie (colonies de fourmis,...). En plus de leur aspect stochastique, les métaheuristiques sont généralement itératives, c.-à-d. qu'un même schéma de recherche est appliqué plusieurs fois au cours de l'optimisation, et d'une manière directe, c.-à-d. qu'elles n'utilisent pas l'information du gradient de la fonction

objectif. Ainsi, l'un des points clés de ces approches et leur capacité à réaliser un équilibre entre la *diversification* et l'*intensification* de la recherche.

D'un côté, la diversification vise à échantillonner l'espace de recherche de façon large de sorte que la recherche ne soit pas concentrée sur une zone particulière de l'espace de recherche. De l'autre côté, l'intensification a pour objectif d'augmenter l'effort de recherche dans les zones les plus prometteuses, aux alentours des bonnes solutions, pour ainsi atteindre la solution optimale. De plus, les métaheuristiques tirent en particulier leur intérêt de leur capacité relative à éviter les optimums locaux, soit en acceptant une dégradation de la fonction objective au cours de leur progression, soit en utilisant une population de points comme méthode de recherche.

Dans le présent mémoire, nous allons travailler principalement sur deux types d'algorithmes métaheuristiques, dont le but principal est de faire introduire des améliorations sur l'un de ces deux. Le premier algorithme est dit ; algorithme de pollinisation des fleurs (en anglais ; Flower Pollination Algorithme- FPA) [1] et le second représente une version simple d'algorithmes génétiques (en anglais ; Genetic Algorithms - GAs). En réalité, l'intérêt de ce travail est de faire optimiser le fonctionnement du FPA en utilisant un algorithme GA simple, en intégrant quelques opérateurs de GA dans le corp algorithmique du FPA. Il est à noter que, l'approche suivie pour réaliser l'objectif visée (amélioration) du FPA, est très utilisée par l'ensemble des chercheurs travaillant dans le domaine, elle porte couramment le nom ; *hybridation*.

L'algorithme de pollinisation des fleurs (FPA) proposé par le Prof. Xin-She-Yang de l'université britannique Middlesex en 2012 [1], est un algorithme d'optimisation inspiré du processus physiologique de reproduction chez les plantes et plus précisément les fleurs. En effet, cet algorithme imite la reproduction des plantes de même genre ou autre, par le biais du processus de pollinisation.

Les algorithmes génétiques (Genetic Algorithms GAs) [2], présente un type spécifique d'algorithmes évolutionnaires. Généralement, leur but est d'obtenir une solution approchée à un problème d'optimisation, lorsqu'elle n'existe pas de méthode exacte pour le résoudre en un temps raisonnable. Les GAs utilisent la notion de sélection naturelle et l'appliquent à une population de solutions potentielles au problème donné.

Le présent mémoire est structuré autour de cinq chapitres ;

Le premier chapitre représente une petite introduction descriptive sur les métaheuristiques à population de solutions dédiées aux problèmes d'optimisation combinatoire mono-objectifs au sens général.

Dans le second chapitre, nous présentons les fondements théoriques de base de l'algorithme FPA.

Le troisième chapitre a le même objectif que le second, il contient des informations théoriques générales sur les AGs,

Le principe de la nouvelle méthode proposée dans ce mémoire sera présenté dans le quatrième chapitre,

Le chapitre cinq expose la validation-au sens général c.-à-d. ; tests, résultats plus analyses- de la méthode développée.

En fin, nous terminerons ce mémoire par une conclusion générale.

Chapitre 1

CHAPITRE 1

Métaheuristiques à population de solutions : Généralités

1.1 Introduction :

La recherche de solution optimale d'un problème constitue une préoccupation importante dans le monde actuel, qu'il s'agisse d'optimiser à titre d'exemple le temps, le confort, la sécurité, les coûts ou les gains. En effet, de nombreux problèmes scientifiques, sociaux-économiques et techniques contiennent des paramètres devant être ajustés pour donner un résultat souhaitable. Ceux-ci peuvent être considérés comme des problèmes d'optimisation et leur résolution représente un sujet principal en recherche opérationnelle.

Plusieurs techniques et méthode ont été proposées pour résoudre ce genre de problèmes, spécialement les problèmes dits « difficiles », en déterminant des solutions qui ne sont pas exclusivement optimales, mais qui s'en approchent. Ces méthodes, appelées heuristiques et métaheuristiques.Elles sont généralement inspirées de métaphores biologiques (Algorithmes Evolutionnaires), d'évolution culturelle des populations (Algorithmes Culturels [3]) ou du comportement collectif des d'*espèces vivantes* spécifiques (insectes sociaux, oiseaux migrateurs, poisson,..).

Dans ce premier chapitre nous présentons une petite introduction descriptive sur les métaheuristiques à population de solutions dédiées aux problèmes d'optimisation combinatoire mono-objectifs au sens général.

Vue la bibliographie, on peut distinguer généralement, deux grandes familles de métaheuristiques : celles basées sur l'évolution itérative d'une solution unique et celles qui manipulent en parallèle toute une population de solutions.

Les approches de la première famille, commencent toujours leur processus de recherche avec une seule solution initiale et s'en éloignent progressivement, en construisant une trajectoire dans l'espace de recherche. Ces méthodes englobent essentiellement la méthode de descente [4], la méthode du recuit simulé [5], la méthode de recherche avec tabous [6], la méthode de recherche gloutonne aléatoire adaptative (*Greedy Randomized Adaptive Search Procedure* GRASP) [7,8], La recherche à voisinage variable [9, 10], la recherche locale itérée [11], ainsi que leurs variantes.

Contrairement aux algorithmes cités ci-dessus, les métaheuristiques à population de solutions améliorent, successivement au cour des différentes itérations, une population de solutions. On trouve dans cette catégorie, les Algorithmes Evolutionnaires, qui sont une famille d'algorithmes issus de la théorie de l'évolution par la sélection naturelle et les algorithmes d'intelligence en essaim qui, de la même manière que les algorithmes évolutionnaires, proviennent d'analogies avec des phénomènes biologiques naturels.

1.2 Métaheuristiques pour l'optimisation mono-objectif difficile

1.2.1 Problème d'optimisation mono-objectif :

Un problème d'optimisation mono-objectif au sens général est défini comme la recherche, parmi un ensemble de solutions possibles S(appelé aussi espace de décision ou espace de recherche), de la (ou des) solution(s) x^*qui rend (ent) minimale (ou maximale) une fonction mesurant la qualité de cette solution. Cette fonction dite habituellement fonction objectif ou fonction coût.

Si l'on pose $f: S \rightarrow \mathbb{R}$la fonction objectif à minimiser (respectivement à maximiser) à valeurs dans $\mathbb{R}$, le problème revient alors à trouver l'optimum $x^* \in S$tel que $f(x^*)$ soit minimal (respectivement maximal).

La résolution d'un problème d'optimisation concerne principalement la recherche de la meilleure solution possible à ce problème, c.-à-d. l'optimum global (en anglais ; *the global optimum*). Néanmoins, il peut exister des solutions intermédiaires, qui sont également des optimums, mais uniquement pour un sous-espace étroit de l'espace de recherche : on parle ici d'optimums locaux.

Cette notion est illustrée dans la figure 1.1. La seule hypothèse faite sur Sest qu'il s'agit d'un espace topologique, c.-à-d. Sur lequel est définie une notion de voisinage. Cette hypothèse est nécessaire pour définir la notion de solutions locales du problème d'optimisation. On peut alors définir un optimum local (relativement au voisinage$V$) comme la solution x^*de S telle que$f(x^*) \leq f(x);\ \forall\, x \in V(x^*)$.

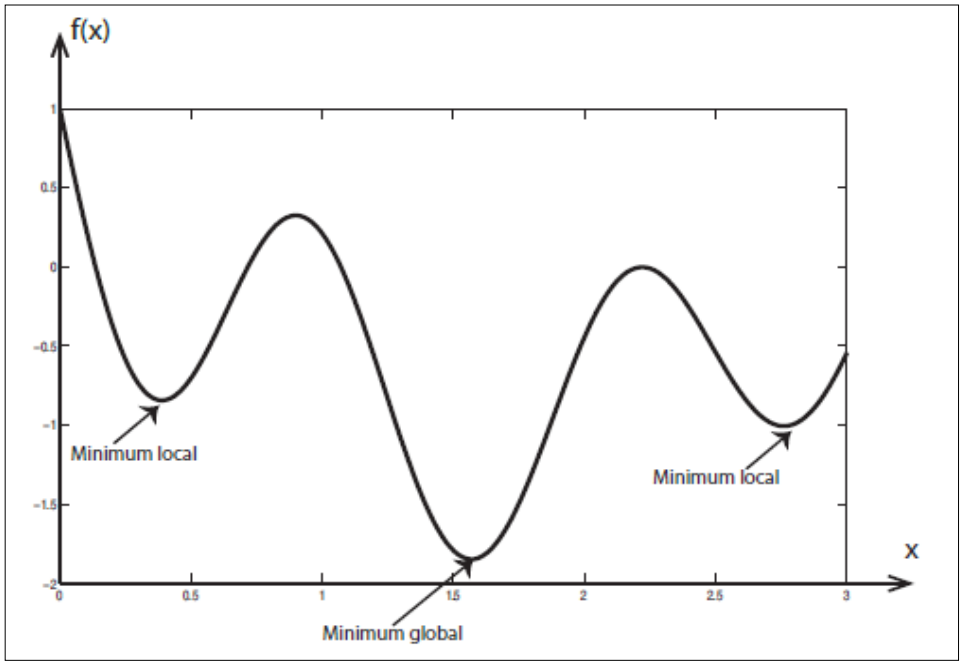


Figure 1.1 : Différence entre un optimum global et des optima locaux [12].

1.2.2 Optimisation difficile :

En réalité pratique, les méthodes d'optimisation déterministes (dite aussi exactes) ne sont pas adaptées à toutes les problématiques, et donc certains problèmes sont excessivement complexes à résoudre par ce genre de méthodes. Parmi ces problématiques, nous pouvons citer l'existence de discontinuités, l'absence de convexité stricte (multi modalité), la non-dérivabilité, la présence de bruit ou encore la fonction objectif peut ne pas être définie précisément. De plus, ces méthodes peuvent avoir un temps de résolution trop long. Dans ce cas, le problème d'optimisation est dit difficile, car aucune méthode exacte n'est capable de le résoudre en un temps raisonnable. Il est alors nécessaire d'avoir recours à des heuristiques de résolution dites méthodes approchées, qui fournissent un résultat sans garantie de l'optimalité.

L'optimisation difficile peut se diviser en deux catégories : les problèmes à variables discrètes et les problèmes à variables continues. D'une façon générale, un problème d'optimisation à variables discrètes, dit aussi combinatoire consiste à trouver, dans un ensemble discret, une solution réalisable. Le problème majeur réside ici dans le fait que le nombre de solutions réalisables est généralement très élevé, donc il est très difficile de trouver la meilleure solution dans un temps « raisonnable ou acceptable ». Les problèmes à variables discrètes rassemblent les problèmes de type NP-complets (*Non-deterministic Polynomial time* NP-complets ; Problèmes Non-déterministes Polynomiaux-complets), tels que le problème du voyageur de commerce (*Traveling Salesman Problem*TSP[13, 14]).

L'expérience à montrer que l'utilisation d'algorithmes d'optimisation stochastique, tels que les métaheuristiques, permet de trouver une solution approchée en un temps raisonnable.

Dans le second type, les variables du problème d'optimisation sont continues. C'est le cas p. ex. des problèmes d'identification, où l'on cherche à minimiser l'/les erreur(s) entre la/les sortie(s) du système réel et celle/celles du modèle modélisant (identifiant) ce dernier. Ce type de problèmes est moins formalisé que le précédent. En effet, la grande majorité des métaheuristiques existantes ont été créées pour résoudre des problèmes à variables discrètes [12]. Cependant, la nécessité croissante de méthodes pouvant adapter avec ce type de problèmes a poussé les chercheurs à adapter leurs méthodes au cas continu. Les difficultés les plus rencontrés avec ce type de problèmes sont généralement les corrélations non identifiées entre les variables, le caractère bruité des fonctions ou encore des fonctions objectifs qui ne sont accessibles que par dispositif expérimental.

Il est à noter aussi, qu'il existe des problèmes à variables mixtes (c.-à-d. Le problème présent à la fois des variables discrètes et continues) [15].

1.2.3 Algorithmes d'optimisation approchée

1.2.3.1 Heuristiques :

L'expérience à montrer que, l'utilisation de méthodes déterministes n'est pas toujours possible pour un problème donné à cause d'un certain nombre de contraintes, à savoir ; le temps de calcul souvent long (n'est pas pratique) ou bien la difficulté, voire l'impossibilité dans certains cas, d'une définition opposée du problème. Pour surmonter ces contraintes, des méthodes spécifiques dites : heuristiques ont été utilisées. En effet, le mot heuristique dérive du grec ancien *heuriskêin* qui signifie « trouver ». Il qualifie tout ce qui sert à la découverte et à l'exploitation. Il est à souligner ici qu'une méthode heuristique peut être déterministe ou stochastique.

Une heuristique, ou méthode approximative, est un algorithme qui fournit rapidement (en temps raisonnable) une solution réalisable, pas nécessairement optimale, pour un problème d'optimisation difficile. Au contraire des méthodes exactes, qui trouvent toujours l'optimum si on leur en laisse le temps (énumération complète, méthodes arborescentes, programmation dynamique), les approches heuristiques, contournent le problème de l'explosion combinatoire [15] en faisant délibérément des impasses et n'explorent qu'une partie de l'espace des combinaisons. Il est à noter aussi que, une méthode heuristique est généralement conçue pour un problème particulier, en s'appuyant sur sa structure propre.

1.2.3.2 Métaheuristiques :

Sans de profonds changements dans leurs structures algorithmiques, plusieurs heuristiques plus développées, adaptables à un grand nombre de problèmes différents, ont été mises au point et ont donné naissance à une nouvelle famille d'algorithmes d'optimisation stochastiques : les

Métaheuristiques. Le terme métaheuristique a été inventé par Fred Glover en 1986, lors de la conception de la recherche tabou [6].

Les métaheuristiques forment une famille d'algorithmes d'optimisation cherchent à résoudre des problèmes d'optimisation difficile, pour lesquels les méthodes classiques ne fonctionnent pas. Elles sont généralement utilisées comme des méthodes génériques pouvant optimiser une large gamme de problèmes différents, d'où le qualificatif méta. En effet, leur capacité à optimiser un problème à partir d'un nombre minimal d'informations est contrebalancée par le fait qu'elles n'offrent aucune garantie quant à l'optimalité de la meilleure solution trouvée. Néanmoins, du point de vue de la recherche opérationnelle, ce constat n'est pas forcément un désavantage, puisque l'on préfère toujours une approximation de l'optimum global trouvée rapidement à une valeur exacte trouvée dans un temps rédhibitoire.

Il existe une grande variété de métaheuristiques, allant d'une simple recherche locale à des algorithmes complexes de recherche globale. La plupart des métaheuristiques utilisent des processus aléatoires comme moyens de rechercher l'information et de faire face à des problèmes comme l'explosion combinatoire [16].

Les métaheuristiques peuvent être considérées comme des algorithmes aléatoires itératifs, où elles manipulent une ou plusieurs solutions à la recherche de l'optimum global. Les itérations successives doivent permettre de passer progressivement d'une mauvaise solution à la solution optimale. L'algorithme s'arrête après avoir atteint un critère d'arrêt, consistant généralement en l'atteinte du temps d'exécution imparti ou en une précision demandée. Ces méthodes tirent leur efficacité du fait qu'elles sont moins facilement piégeables dans des optima locaux, car elles acceptent, au cours du traitement, des dégradations de la fonction objectif et la recherche est souvent menée par une population de points et non un point unique.

Les métaheuristiques sont habituellement conçues pour résoudre des problèmes de nature discrète c.-à-d. à variables discrètes, mais font de plus en plus l'objet d'adaptations aux problèmes à variables continues. De plus, de par leur variété et leur capacité à résoudre des problèmes très divers, les métaheuristiques sont assez facilement sujettes à extensions. Parmi celles-ci, on trouve des métaheuristiques pour : optimisation multi-objectif, optimisation multimodale, optimisation de problèmes bruités, optimisation dynamique, …etc.

Le monde des métaheuristiques est un monde en constante évolution. Un grand nombre de méthodes (soit nouvelles ou améliorées) sont proposées chaque année pour tenter d'améliorer La résolution des problèmes les plus complexes. Du fait de cette activité permanente, un grand nombre de classes de métaheuristiques existe actuellement.

1.3 Métaheuristiques à population de solutions issues de la théorie de l'évolution (Algorithmes Evolutionnaires) :

Les algorithmes évolutionnaires (*Evolutionary Algorithms*EAs) sont des techniques de recherche inspirées principalement de l'évolution biologique des espèces, ils ont apparu à la fin des années 1950. Ce genre d'algorithmes, se basent essentiellement sur les grands principes rencontrés dans la nature et particulièrement celui de l'évolution des espèces et de la sélection naturelle énoncés par Charles Darwin [17].

Un algorithme évolutionnaire a un principe très simple. Par définition, un ensemble de Npoints dans un espace de recherche, choisi a priori au hasard, constituent la population initiale ; chaque individu x de la population possède une certaine performance, qui mesure son degré d'adaptation à l'objectif visé : dans le cas de la minimisation p. ex. d'une fonction objectifF_{obj}, x est d'autant plus performant que $F_{obj}(x)$ est plus petit. Un EA consiste à faire évoluer graduellement, par générations successives, la composition de la population, en maintenant sa taille constante. Au cours des générations, l'objectif est d'améliorer globalement la performance des individus ; le but étant d'obtenir un tel résultat en imitant les deux principaux mécanismes qui régissent l'évolution des êtres vivants, selon la théorie de Darwin :

– la sélection, qui favorise la reproduction et la survie des individus les plus performants,

– la reproduction, qui permet le brassage, la recombinaison et les variations des caractères héréditaires des parents, pour former des descendants aux potentialités nouvelles.

Selon la façon de l'emploi et de la présentation des deux principes Darwiniens cités ci-dessus dans le modèle artificiel, quatre approches différentes ont été proposées [17] : les algorithmes génétiques (*Genetic Algorithms*GAs) [18], la programmation génétique (*Genetic Programming*GP) [19,20], les stratégies d'évolution (*Evolutionary Strategy*ES)[21]et la programmation évolutive (*Evolutionary Programming*EP) [22, 23].

Dans ce travail, nous allons décrire les fondements de base des algorithmes génétiques, puisque nous l'utiliserons dans le développement de notre nouvelle variante.

De plus, plusieurs autres modèles d'algorithmes évolutionnaires ont été proposés dans la littérature [13].Parmi les plus connus dans la littérature, nous nous intéressons principalement à l'algorithme FPA.Ce dernier est considéré comme un algorithme évolutionnaire.

La structure générale d'un algorithme évolutionnaire est donnée par le pseudo code (Algorithme 1.1) illustré ci-dessous :

Algorithme 1.1 Structure de base d'un algorithme évolutionnaire [24]

$\boldsymbol{t \longleftarrow 0}$
Initialiser la population $\boldsymbol{P(t)}$
Evaluer $\boldsymbol{P(t)}$
Répéter
$\boldsymbol{t \longleftarrow t + 1}$
Sélectionner les parents
Appliquer les opérateurs génétiques
Evaluer la population des enfants crées
Créer la nouvelle population $\boldsymbol{P(t)}$ en utilisant une stratégie de sélection
Tant que (condition d'arrêt n'est pas satisfaite)

1.4 Métaheuristiques à population de solutions issues de l'intelligence en essaim :

L'intelligence en essaim (*Swarm Intelligence* SI) est née de la modélisation mathématique et informatique des phénomènes biologiques rencontrés en éthologie [25]. Elle désigne les capacités cognitives d'une communauté donnée résultant des interactions multiples entre les membres (ou agents) de la communauté. Des agents, au comportement très simple, peuvent ainsi accomplir des tâches complexes grâce à un mécanisme fondamental appelé *synergie*. Sous certaines conditions particulières, la synergie créée, par la collaboration entre individus, fait émerger des possibilités de représentation, de création et d'apprentissage supérieures à celles des individus isolés.

Les formes d'intelligence en essaim sont très diverses selon les types de communauté et les membres qu'elles réunissent. Tout système basé sur l'intelligence en essaim a les caractéristiques principales suivantes :

1. L'information locale : Chaque individu/membre il n'a qu'une connaissance partielle de l'environnement c.-à-d. il n'a pas conscience de la totalité des éléments qui influencent le groupe,
2. L'ensemble de règles : Chaque individu obéit à un ensemble restreint de règles simples par rapport au comportement du système global,
3. Les interactions multiples : Chaque individu est en relation avec un ou plusieurs autres individus du groupe,
4. La collectivité : les individus trouvent un bénéfice à collaborer (parfois instinctivement) et leur performance est meilleure que s'ils avaient été seuls.

L'intelligence en essaim est observée notamment chez les insectes sociaux (fourmis, termites, abeilles,…) et les animaux en mouvement (oiseaux migrateurs, bancs de poissons,..).

Plusieurs algorithmes basés sur le principe d'intelligence en essaim ont été introduits, parmi les plus célèbres figurent : l'algorithmes d'optimisation par essaim particulaire (*Particle SwarmOptimization*PSO) [26] qui s'inspire du comportement social des animaux évoluant en essaim, tels que les nuées d'oiseaux et les bancs de poissons, l'algorithme *Bacterialforagingoptimization*BFO [27] inspiré du comportement de recherche de nourriture et de reproduction des bactéries, l'optimisation par colonie d'abeilles (*Bee Colony Optimization*BCO) [28] et l'algorithme de colonies de fourmis ACO (*Ant colony optimization* ACO) [29, 30, 31].

1.5 Conclusion :

Dans ce chapitre, nous avons présenté une petite introduction sur les métaheuristiques à population de solutions dédiées aux problèmes d'optimisation combinatoire mono-objectifs. Tout d'abord, nous avons présenté respectivement les définitions : d'un problème d'optimisation mono-objectif, de l'optimisation difficile et des algorithmes d'optimisation approchée (heuristiques et métaheuristiques). Puis, nous avons présenté les deux types de métaheuristiques à population de solutions, celles issues de la théorie de l'évolution (algorithmes évolutionnaires) et les autres issues de l'intelligence en essaim.

Dans le chapitre suivant, nous présentons quelques fondements de base des AGs, puisque ils utilisé par la suite en hybridation avec l'algorithme FPA afin de créer une nouvelle variante a ce dernier.

Chapitre 2

CHAPITRE 2

Algorithme de pollinisation (Flower Pollination Algorithme-FPA)

2.1 Introduction :

Le présent chapitre a pour but de présenter l'algorithme de pollinisation (en anglais ; Flower Pollination Algorithme). En effet, ce dernier est basé principalement sur la modélisation du processus de pollinisation (au sens général) des fleurs dans la nature. Un aperçu général sur les fondements théoriques de base de cet algorithme sera présenté.

2.2 Algorithme de pollinisation (Flower Pollination Algorithme-FPA) :

L'algorithme de pollinisation des fleurs (FPA) proposé par le Prof. Xin-She-Yang de l'université britannique Middlesex en 2012 [1], est un algorithme d'optimisation (métaheuristique à population de solutions) inspiré du processus physiologique de reproduction chez les plantes et plus précisément les fleurs. En effet, cet algorithme imite la reproduction des plantes de même genre ou autre, par le biais du processus de pollinisation. Pour mieux comprendre le principe de cette méthode d'optimisation, nous commençons par donner une brève description de ses fondements biologiques de base [1].

2.2.1 Fondements biologiques de l'algorithme FPA :

Généralement, il est porté à la connaissance de tout le monde que la reproduction de presque toutes les plantes, au son sens simple et direct, est le résultat d'une opération de pollinisation. De ce fait, ce processus biologique naturel très important est généralement associé au transfert d'un substance chimique appelé pollen, lié à certaines créatures appelées pollinisateurs tels que les insectes, les oiseaux, les chauves-souris et autres. En effet, certain type de pollinisateurs et certain type de fleurs sont réunis pour créer une sorte de coopération fleurs-pollinisateurs très spécifiques.

Par exemple, certains types de pollinisation ne peuvent être réalisés sans l'intervention d'un type spécifique de pollinisateurs. En réalité, le processus de pollinisation comporte deux formes principales : la pollinisation biotique et celle abiotique. Ainsi, environ 90% des plantes à fleurs appartiennent à la première classe, dans laquelle le pollen est transféré par un pollinisateur d'un type spécifique. À propos de la seconde classe qui représente un pourcentage de 10%, le

processus de pollinisation est effectué principalement par des facteurs naturels tels-que ; le vent, l'eau ou la gravité.

Les éléments pollinisateurs, ou parfois appelés vecteurs pollen, peuvent être représentés par différentes espèces comme p.ex. Les abeilles domestiques, ce dernier (pollinisateur) représente un facteur principal dans le cas de la pollinisation de type biotique. Il à noter que, certain type de pollinisateur sont tendance de visiter exclusivement une espèce de fleur ; Ce comportement est appelé constance florale (Flower Constancy). Par le fait, cette dernière (constante) fait l'augmentation directe de la reproduction chez le même type de fleur en maximisant le transfert de pollen vers les mêmes plantes.

Selon la disponibilité des pollinisateurs, deux types de pollinisation sont considérés : *Auto pollinisation* et la *Pollinisation croisée*. Le premier type, appelé aussi *Pollinisation locale (Self-Pollination)*, se produit lorsqu'il s'agit d'une pollinisation à partir (source) d'une fleur vers d'autres appartenant à la même plante [1]. A l'opposé, la pollinisation croisée, également appelée *Pollinisation globale (Cross-Pollination)*, elle est produite lorsque le pollen est transféré d'une fleur appartient à une plante placée à distance par rapport à autre et ceci à travers une intervention directe ou indirecte des éléments pollinisateurs qui obéissent à un régime de vol (modélisé) dit ; vol de Lévy (Lévy flight) [1] (Voir figure 2.1)

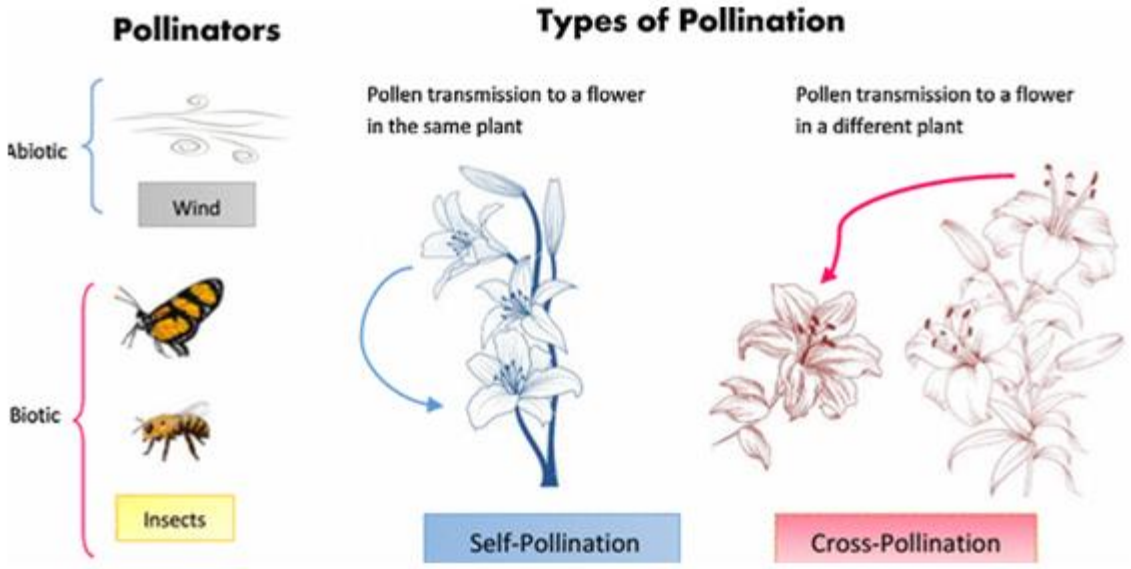


Figure 2.1 : Eléments pollinisateurs et types de pollinisation.

2.2.2Algorithme FPA ; principe :

Le Prof. Xin-She-Yang à créer l'algorithme FPA basant sur les fondements théoriques présentés préalablement, son algorithme est basé les quatre règles principales suivantes :

Règle 1 ;

Le processus de pollinisation global a lieu à travers la pollinisation biotique et croisée, sachant que les pollinisateurs ont un mouvement obéisse à un régime de vol de Lévy (Lévy flight) [1],

Règle 2 ;

Le processus de pollinisation locale est considéré comme abiotique et auto pollinisation,

Règle 3 ;

La constance florale (Flower Constancy) fournie par les éléments pollinisateurs ; est équivalent à une probabilité de reproduction proportionnelle à la ressemblance de deux fleurs impliquées dans le processus pollinisation,

Règle 4 ;

L'orientation du processus global de pollinisation, soit vers la pollinisation locale ou globale est contrôlée par un paramètre de commutation (variable aléatoire) $p \in [0,1]$avec une simple orientation vers la pollinisation locale pour des raisons liées à l'approximation de l'algorithme au cas réel.

L'implémentation de ces règles est basée sur une idée qui dite que : chaque plante n'a qu'une fleur, et chaque fleur ne produit qu'un seul gamète de pollen [1]. De plus, cet argument signifie qu'il n'est pas nécessaire de distinguer entre un gamète de pollen, une fleur, une plante ou une solution à un problème donné.

Le passage à la formulation mathématique de ces quatre règles est effectué conformément à [1] comme suit :

Au début, le processus de pollinisation globale (règle 1) et la constance florale (Règle 3) sont représentés à l'aide de l'équation suivante :

$$x_i^{t+1} = x_i^t + \gamma L(\lambda)(g_* - x_i^t) \tag{2.1}$$

D'où ; x_i^test le pollen i ou le vecteur de solution $\boldsymbol{x_i}$ à l'itération t, x_i^{t+1} est le vecteur solution générée à l'itération$t+1$, g^*est la meilleure solution actuelle. γ c'est un facteur d'échelle utilisé pour contrôler la taille de l'étape. $L(\lambda)$ représenté la fonction modélisant le vol de Lévy. En réalité, les pollinisateurs peuvent survoler une longue distance avec des pas de distance différents cela peut être modélisé en utilisant une répartition de Lévy [1] selon l'équation suivante :

$$L \sim \frac{\lambda\Gamma(\lambda)\sin\left(\frac{\pi\lambda}{2}\right)}{\pi}\frac{1}{s^{1+\lambda}}(s \gg s_0 \gg 0) \tag{2.2}$$

Dans cette équation, Γ(λ)représente la fonction gamma standard, et cette distribution est valable pour les grandes étapes $s > 0$.Ensuite, la pollinisation locale (règle 2) et la constance des fleurs (Flower Constancy)(règle 3) peuvent être représentés comme suit :

$$x_i^{t+1} = x_i^t + \epsilon\left(x_j^t - x_k^t\right) \tag{2.3}$$

D'où x_j^tet x_k^t sont des gamètes de pollen obtenus à partir de fleurs de la même espèce végétale. De plus, cette soustraction aléatoire $(x_j^t - x_k^t)$ est utilisée pour imiter la constance des fleurs dans un voisinage limité. Le paramètre ϵ est choisi arbitrairement dans [0,1] pour rapprocher cette sélection à une valeur locale.

Le processus de pollinisation est déroulé sous une permutation aléatoire entre le cas local et le cas global. Par conséquent, pour contrôler cette double-orientation aléatoire, le paramètre $p \in [0,1]$dit de commutation (règle 4) est utilisé.

L'algorithme FPA est illustré ci-dessous :

Algorithme 2.1 Pseudo-code de l'algorithme FPA standard [1]

1: Définir une fonction objectif $f(x), x = (x_1, \ldots., x_d)^T$
2 : Générer une population de n_f fleurs dans des positions aléatoires
3 : Trouer la bonne solution g_* dans la population générée initialement
4 : Calculer le paramètre$p \in [0, 1]$
5 : Initialiser le compteur d'itération $t = 0$
6 : **Tanque** t<t_{max} **faire**
7: **Pour** $i = 1 : n_f$(Toutes les$n_f$fleurs de la population) **faire**
8: **Si** Nombre aléatoire ∈ [0, 1]<$p$alors
9: Dessiner une voie basée sur la fonction Lévy
10: Réaliser une pollinisation globale à travers l'équation (2.1)
11: **Sino**
12 : Choisir un paramètre ϵaléatoirement à partir de l'intervalle [0,1]
13 : Choisir x_j^tetx_k^taléatoirement de la population
14 : Réaliser une pollinisation locale à travers l'équation (2.3)
15 : **Fin Si**
16 : Evaluer le nouveau vecteur de solution x_i^{t+1}
17 : Si le nouveau vecteur généré est meilleur que le précédent, replace x_i^t by x_i^{t+1}
18 : **Fin Pour**
19 : Mise-à-jour (renouveler)g^*
20 : $t = t + 1$
21 : **Fin Tanque**

2.3 Quelques variantes de l'algorithme FPA :

Bien que l'efficacité de l'algorithme FPA est prouvée, il présente comme toutes les autres techniques appartenant à sa famille, certains inconvénients tels que ; le temps de calcul (relativement long) et la convergence prématurée pour un certain nombre de problèmes

donnés. En effet, plusieurs améliorations ont été proposées pour sur monter principalement ces deux inconvénients. L'ajustement de l'algorithme peut être fait au niveau de la recherche globale ou locale ou toutes les deux en même temps.

Nous citons dans ce qui suit quelques petits exemples (un ou deux) relatifs aux variantes les plus célèbre de l'algorithme FPA basées sur les facteurs suivants ; (i) Amélioration, (ii) Adaptation à l'espace de recherche et finalement (iii) l'hybridation ;

A. Variantes développées basées sur l'introduction d'une amélioration spécifique ;

1. *Modified FPA (MFPA)* :

Dont le but d'améliorer la convergence de l'algorithme FPA (version de base) Dubey et ses collègues (Voir [32]) ont proposé en (2015) une nouvelle variante de FPA dite MFPA, basée sur le remplacement du paramètre ϵ (aléatoire) qui contrôle la pollinisation locale par un nouveau facteur d'échelle constant. De plus, une étape d'exploitation supplémentaire a été ajoutée pour ajuster la meilleure solution. La variante MFPA a été utilisée pour résoudre des problèmes de répartition économique relative aux systèmes d'alimentation électrique (Economic Dispatch Problems of ElectricalPower Systems). Les résultats expérimentaux ont montré que l'efficacité des modifications introduites.

2. *Improved FPA with Chaos (IFPCH):*

Les cartes chaotiques (Chaoticmaps) (carte logistique, carte de Chebyshev, carte de tente, etc.) sont des fonctions d'évolution, elles permettent la génération d'une séquence déterministe et bornée de nombres aléatoires en fonction de la condition initiale avec des différents domaines temporels (continu ou discret). Abdel-Raouf et ses collègues (Voir [33]) ont utilisé ce genre de cartes au lieu des nombres aléatoires, en exploitant leurs formes aléatoires, non-période, et non convergent pour une adaptation des paramètres pour la création d'une nouvelle variante dite ; IFPCH. En effet, cette nouvelle variante a une probabilité de commutation, un pas de Lévy et un nombre aléatoire de pollinisation locale définis en fonction de la carte chaotique sélectionnée.

L'algorithme proposé est implémenté (simulé) afin de calculer la valeur numérique des intégrales définies. Les résultats obtenus, ont montré que l'ajout de cartes chaotiques au FPA avait un effet significatif sur ses performances.

B. Variantes développées basées sur le facteur adaptation à l'espace de recherche ;

Comme l'algorithme FPA (version de base) a prouvé son efficacité pour des problèmes d'optimisation mono-objectif de type contenu, plusieurs études ont été réalisée dont le but est de faire étendre ce dernier à d'autres environnements de recherche (d'autres types d'espace de recherche). Dans ce qui suit, on présente deux variantes de l'algorithme FPA adaptées avec des espaces de recherche de types spécifiques.

1. *Espace combinatoire (Combinatorialspace)* :

Rodrigues et ses collègues (Voir [34]) ont proposé une variante relative au FPA adaptée avec les problèmes de type binaire discret, où l'espace de recherche est représenté par un réseau ou treillis de type booléen de *d*-dimension. Afin de faire un transfert d'un espace de recherche de type continu vers un autre de type binaire une fonction de type sigmoïde (SF) est utilisée comme suit:

$$x_{ij} = f(x) = \begin{cases} 1, & rand() \leq \frac{1}{1+exp(-x_{ij})} \\ 0, & otherwise \end{cases} \quad (2.4)$$

Également, Bensouyad et Saidouni (Voir [35]) ont utilisé une fonction d'arrondissement pour l'obtention d'un vecteur de solution composé de nombres entiers. De plus, Mishra et Deb (Voir [36]) ont aussi proposé une autre méthode de discrétisation basée sur la représentation des vecteurs de solutions sous une forme de chaînes de nombres dont chacun est composé de deux chiffres.

2. *Optimisation multi-objectif (Multi-objective optimization)* :

Une version d'algorithme FPA de type multi-objectif a été proposée par Yang et ses collègues (Voir [37] et [38]). Les auteurs ont proposé méthode afin de transformer le problème traité de type multi-objectif en un de type mono-objectif.

B. Variantes développées basées sur l'hybridation entre FPA et autres méthodes ;

Une autre façon considérée très efficace pour l'amélioration des performances d'un algorithme ou de combler certaines de ses lacunes consiste à le combiner avec une autre méthode. Ce principe général, appelé hybridation, peut s'appliquer pour un grand nombre de méthodes. L'algorithme FPA ne fait pas exception à la règle, et une grande multitude d'algorithmes FPA hybrides ont fait leur apparition ces dernières années. Nous pouvons notamment citer les exemples suivant (le lecteur peut se référer aux références pour prendre plus d'informations) ;

Nabil et ses collègues, Ont proposé une variante de FPA hybride basée sur la combinaison entre le processus de pollinisation locale à la méthodologie de sélection clonale [39]. De plus, récemment une nouvelle variante de l'algorithme FPA a été proposée par Kalra and Arora [40], ce dernier est basée sur l'hybridation entre la version FPA standard et l'algorithme Firefly Algorithme (FA). Dans le même stade de recherche, Yang et ses collègues ont hybridé la technique nommée ; Egale Strategy (ES) [41] et la partie du FPA relative à la recherche locale afin de créer un nouvel algorithme a une exploitation et une intensification meilleures que celles de version de FPA standard.

Dans tous les cas, le lecteur a la possibilité de découvrir d'autres variantes hybrides de FPA an consultant les références suivantes ; [42].

2.4 Conclusion :

Dans présent chapitre, nous avons présenté les notions de base de FPA, à savoir ; ses fondements biologiques sur lesquelles le Prof. Xin-She-Yang basé pour la création de ce dernier. En outre, la version imitée de l'algorithme FPA (version algorithmique) est présentée avec des explications intensives relatives aux quatre règles sur lesquelles repose cet algorithme ainsi les équations qui gèrent son fonctionnement. Au demeurant, et comme toutes les métaheuristiques, le FPA a connu des améliorations significatives introduites sur sa structure. A cet effet, une partie complète de ce chapitre est concrétée à ce sujet (amélioration de FPA), chose sur laquelle nous parlerons dans le chapitre suivant. Plus précisément, nous présenterons le développement d'une nouvelle variante de FPA basée sur l'hybridation de ce dernier avec un algorithme GA standard.

Chapitre 3

CHAPITRE 3

Les algorithmes génétiques (Genetic Algorithms)

3.1 Introduction :

Les algorithmes génétiques (Genetic Algorithms GAs), ils appartiennent comme nous l'avons dit précédemment (voir chapitre 1, section 1.3) à la famille des Algorithmes Evolutionnaires. Leur but est d'obtenir une solution approchée à un problème d'optimisation, lorsqu'il n'existe pas de méthode exacte (ou que la solution est inconnue) pour le résoudre en un temps raisonnable. Les GAs utilisent la notion de sélection naturelle et l'appliquent à une population de solutions potentielles au problème donné. La solution est approchée par « bonds » successifs, comme dans une procédure de séparation et évaluation branch & bound, à ceci près que ce sont des formules qui sont recherchées et non plus directement des valeurs.

Dans ce chapitre, nous montrerons quelques principes de base de ce genre d'algorithme, chose que nous permet de faciliter l'implémentation et l'utilisation de ces derniers dans le travail à réaliser.

3.2Les algorithmes génétiques (Genetic Algorithms) ; présentation générale :

La théorie des algorithmes génétiques (*Genetic Algorithms* GAs) a été initialement développée par John Holland en 1960 et a été entièrement élaborée dans son livre *"Adaptation in Natural and Artificial Systems"*, publié en 1975 [18]. Son objectif préalable n'était pas le développement d'un algorithme d'optimisation, mais plutôt la modélisation du processus d'adaptation, et montrer comment ce processus pourrait être utile au sein d'un système de calcul. En effet, les GAssont des techniques de recherche stochastique qui utilisent les concepts de la génétique mendélienne et ceux de l'évolution darwinienne.

Une population d'individus choisis dans l'espace de recherche, souvent d'une façon aléatoire, sert de solutions candidates au problème à optimiser [43]. Les individus de cette population sont évalués grâce à une fonction d'adaptation dite ("fitness"). Un mécanisme de sélection est ensuite utilisé pour sélectionner les individus à utiliser comme parents pour ceux de la prochaine génération. Ces individus seront ensuite croisés puis mutés pour constituer la nouvelle progéniture. La prochaine génération est enfin constituée par un mécanisme de remplacement entre les parents et leur progéniture [44].Ce processus est répété jusqu'à

Satisfaction d'une certaine condition d'arrêt. Ces différentes étapes sont détaillées dans les paragraphes suivants.

a. Représentation des solutions :

Cette étape importante définit la façon dont les individus de la population (solutions candidates) seront codés afin d'être évalués par la fonction objectif. Dans les travaux de Hollande [18] ces solutions sont représentées par des chaînes binaires de longueur fixe. Néanmoins, ce type de codage est très difficile dans le cas de la représentation des solutions de plusieurs problèmes d'optimisation p. ex. en industrie ou en ingénierie [45].Plusieurs types de codage peuvent être utilisés, en fonction du problème, p. ex., binaire, réel, entier ou permutations des éléments. Il est important de noter que, la sélection d'un bon codage est un choix critique dans la conception des AGs [46].

b.Population des solutions candidates :

Une population est formée par un ensemble d'individus, appelés également chromosomes, spécifiquement d'une taille fixe. Chaque individu représente une solution possible au problème à résoudre et consiste en une séquence d'éléments plus petits, appelés gènes. Chaque gène peut prendre des valeurs différentes, ou dites allèles. Dès le début de l'évolution, une population initiale est générée d'une façon aléatoire, cependant il existe plusieurs heuristiques pour créer cette population initiale. Cette population évolue au fil des générations. A la fin du processus, il est prévu qu'un algorithme GA converge vers une région de l'espace de recherche qui contient une solution acceptable. Le paramètre taille d'une population est un élément important dans le paramétrage desGAs, qui influence la complexité de l'algorithme. Typiquement, la taille de la population est constante, mais il existe plusieurs versions des GAsoù cette taille est dynamique [47].

c.*Fonction objectif :*

La fonction *objectif* sert à mesurer la qualité des individus de la population. Cette mesure est soit à maximiser soit à minimiser. Généralement, la définition de cette fonction représente une tâche difficile et une étape très essentielle. Une mauvaise formulation de cette fonction pourra mener les GAs vers des solutions non souhaitables.

d. Sélection:

La sélection [48] est utilisée pour choisir les parents qui vont survivre pour produire les enfants de la prochaine génération. Les parents avec les meilleures valeurs d'adaptation (fonction *objectif*) ont plus de probabilité d'être choisi pour l'accouplement. Ils existent plusieurs méthodes de sélection, parmi elles : la méthode dite roulette biaisée, la sélection par rang, la sélection par tournoi, la sélection de Boltzmannet bien d'autres [49]. La roulette biaisée sélectionne des individus en fonction de leur valeur d'adaptation relative : des individus de plus

grande valeur d'adaptation ont une probabilité plus élevée d'être sélectionnés. Dans la sélection par tournoi, un certain nombre (taille du tournoi) des individus est choisi au hasard, et le meilleur individu est sélectionné. La méthode de sélection par rang est basée sur le rang des individus en fonction de leurs valeurs d'adaptation. Chaque individu a une certaine probabilité d'être choisi, en fonction de son rang.

e. Croisement :

L'opérateur de croisement [50], [51] vise à orienter la recherche vers les régions prometteuses de l'espace de recherche. Pour cela, les individus (enfants) crées par croisement héritent les meilleures caractéristiques de leurs parents et peuvent exploiter au mieux l'espace de recherche. L'exemple le plus simple de ce type d'opérateur est appelé croisement à un point. Il consiste à choisir au hasard un point de croisement pour chaque couple d'individus sélectionné. Ce point divise le génome en deux sous-chaînes. On échange ensuite les deux sous-chaînes terminales de chacun des deux chromosomes, ce qui produit deux enfants. La figure 1.3 illustre le principe de ce type de croisement.

Un autre opérateur de croisement bien connu est appelé croisement uniforme, où les gènes des deux parents sont entrelacées selon un masque aléatoire. On trouve également l'opérateur appelé "*simulated bianry crossover* SBX" qui est une autre version de croisement à un point dans le cas d'un codage réel.

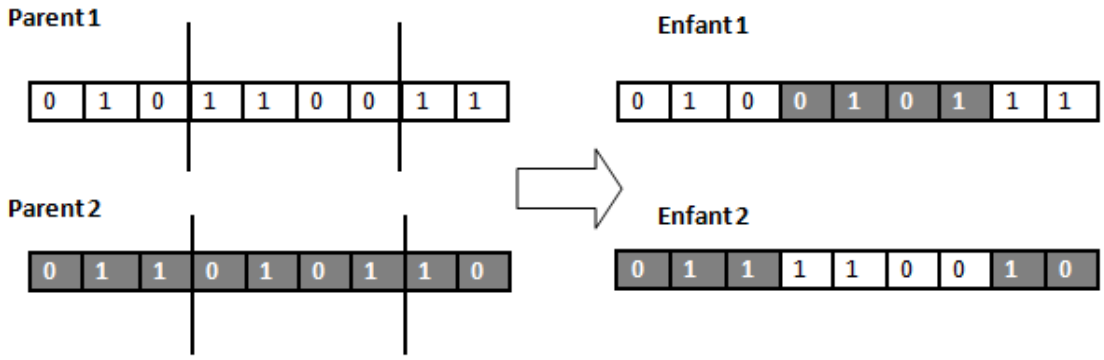


Figure 3.1 : Exemple de croisement en représentation binaire.

f. Mutation :

Le processus mutation [52] consiste à altérer la composition génétique de quelques chromosomes pour créer une diversité génétique au sein de la population. Cet opérateur permet d'explorer de nouveaux espaces de recherches et permet à l'algorithme GA de donner chance à d'autres solutions candidates. La figure 3.2 montre un exemple de mutation pour un individu codé en représentation par permutation. Cependant, les deux opérateurs génétiques, croisement et mutation sont appliqués avec une certaine probabilité. En général une faible probabilité est accordée à la mutation alors que le croisement se voit accordé une forte probabilité. En effet, un

taux de mutation élevé va rendre l'algorithmeGA comme une recherche aléatoire et ne va pas converger vers la solution optimale.

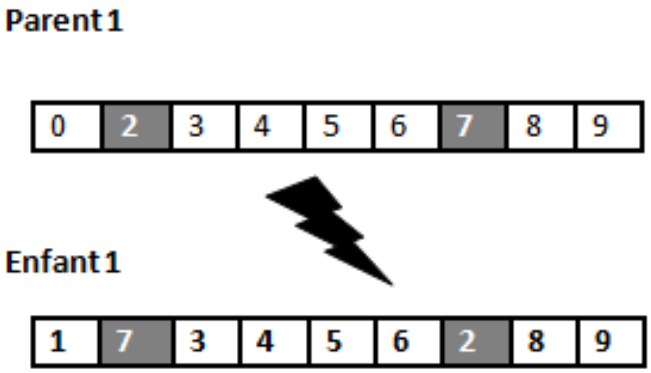


Figure 3.2 : Exemple de mutation en représentation par permutation.

g. Remplacement :

Le remplacement est l'opération qui consiste à passer d'une génération à une autre [48]. Il peut se faire de deux façons : la première consiste à garder uniquement les descendants, c'est l'approche générationnelle ; la deuxième consiste à fusionner les deux populations en choisissant les meilleures de leurs individus (parents et fils). Une version possible de ce fusionnement est de garder une petite fraction des meilleurs parents et de compléter la population par la meilleure progéniture, cette approche est appelée *élitisme* [53].

Algorithme 1.2 Pseudo-code de l'algorithme génétique standard [54]

Entrées : **NP** : Taille de la population ; $\boldsymbol{p_x}$: Probabilité de croisement ; $\boldsymbol{p_m}$: Probabilité de mutation
Sorties : Le meilleur individu
$\boldsymbol{t \leftarrow 0}$
Choisir **NP**individus ($\boldsymbol{I_i}$) d'une façon aléatoire pour former la population $\boldsymbol{P(t)}$
Tant que *Condition d'arrêt≠Vrai* faire
  Évaluer ($\boldsymbol{P(t)}$)
  $\boldsymbol{P_1(t)} \leftarrow$*Selection* ($\boldsymbol{P(t)}$)
  $\boldsymbol{P_2(t)} \leftarrow$*Croisement* ($\boldsymbol{P_1(t)}$)
  $\boldsymbol{P_3(t)} \leftarrow$*Mutation* ($\boldsymbol{P_2(t)}$)
  $\boldsymbol{P(t)} \leftarrow$*Remplacement* ($\boldsymbol{P_1(t)}, \boldsymbol{P_3(t)}$)
  $\boldsymbol{t \leftarrow t + 1}$
Fin

3.3 Conclusion :

Dans ce chapitre, nous avons présenté une quelque fondement de base des algorithmes génétiques GAs. En effet, ces derniers ils appartiennent à la famille des Algorithmes Evolutionnaires. Selon le principe de la version de base (basic version of GAs) leur but est d'obtenir une solution approchée à un problème d'optimisation donné, lorsqu'il n'existe pas de méthode exacte (ou que la solution est inconnue) pour le résoudre en un temps raisonnable.

Par le fait, ce genre d'algorithmes reposent principalement sur quatre opérateurs différents ; à savoir ; la sélection, le croisement, la mutation et le remplacement, chacun de ces opérateur a une importance primordiale.

Dans le chapitre suivant, nous présentons le principe de l'algorithme développé basé sur une hybridation directe entre une version classique des GAs ainsi l'algorithme FPA.

Chapitre 4

CHAPITRE 4

Développement d'une nouvelle variante de FPA ; le FPA-GA

4.1 Introduction :

Afin de surmonter quelques problèmes rencontrés lors de l'utilisation de la version de base de l'algorithme FPA tels que le problème de la convergence prématurée vers des optimums locaux, la convergence lente et la stagnation dans un optimum local, nous proposons dans ce mémoire l'introduction de quelques modifications sur la version standard primitif de FPA. Par le fait, les changements introduits sont basés principalement sur l'emploi d'une approche très utilisée par les chercheurs ; c'est la méthode d'hybridation entre deux algorithmes (même plus) différents dont l'intérêt est de bénéficier des avantages et des capacités de chacun d'entre eux. En réalité, l'hybridation est considérée comme une alternative très efficace comme le montre le nombre des travaux réalisés dans ce domaine.

Dans ce chapitre, nous présentons en détail le développement d'une nouvelle variante d'algorithme FPA, basée principalement sur l'hybridation entre le FPA et un algorithme GA implémenté sous sa forme de base.

4.2 Idée de base :

Comme nous l'avons dit précédemment, le but de cette étude (ou travail) est de faire créer une nouvelle variante de FPA a des capacités améliorées par rapport à la version standard.

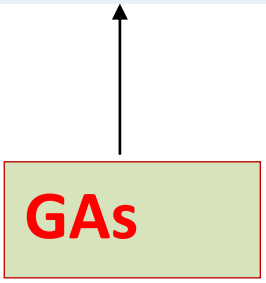
A cet effet, nous avons développé une variante basée sur l'idée de base décrite comme suit ;

L'idée est très simple en terme algorithmique ; Nous avons intégré tout simplement les deux opérateurs d'AGs dits respectivement ; Opérateur de croisement et celui de mutation dans la boucle du programme principal d'un algorithme FPA, juste après le calcul d'un nouveau vecteur de solutions x_i^t (généré à travers une itération t dans un algorithme FPA). Plus précisément, l'utilisation de ces deux opérateurs a pour but de faire renforcer la caractéristique diversification par génération de deux populations supplémentaires (plus de celle courante ou initiale dite ; Pop_t) dites respectivement Pop_{Croos} et Pop_{Mut}.

L'algorithme donné ci-dessous nous permet de bien comprendre le principe de cette nouvelle variante FPA-GA ;

Algorithme 4.1 Pseudo-code de l'algorithme FPA-GA développé

1: Définir une fonction objectif $f(x), x = (x_1, \ldots., x_d)^T$
2: Générer une population de n_f fleurs dans des positions aléatoires
3: Trouer la bonne solution g_* dans la population générée initialement
4: Calculer le paramètre $p \in [0, 1]$
5: Initialiser le compteur d'itération $t = 0$
6: **Tanque**$t < t_{max}$**faire**
7: **Pour** $i = 1 : n_f$(Toutes les n_f fleurs de la population) **faire**
8: **Si** Nombre aléatoire $\in [0, 1]$<p alors
9: Dessiner une voie basée sur la fonction Lévy
10: Réaliser une pollinisation globale à travers l'équation (2.1)
11: **Sino**
12: Choisir un paramètre ϵ aléatoirement à partir de l'intervalle [0,1]
13: Choisir x_j^t etx_k^taléatoirement de la population
14: Réaliser une pollinisation locale à travers l'équation (2.3)
15: **Fin Si**
16: Evaluer le nouveau vecteur de solution x_i^{t+1}
17: Si le nouveau vecteur généré est meilleur que le précédent, replace x_i^t by x_i^{t+1}
18: **Fin Pour**
19: $t = t + n_f$
20: **Si** $t > t_{max}$ alors
21: **Break ;**
22: **Fin Si**
23: **Si** Nombre aléatoire $\in [0, 1]$<$Prob\_Cr\_Mu$ alors
24: Appliquer l'opérateur de croissement sur la population courante $Pop_t \rightarrow Pop_{Cross}$
25: Appliquer l'opérateur de Mutation sur la population courante $Pop_{Cross} \rightarrow Pop_{Mut}$
26: Faire une sélection élite (Elite Sélection) de n_f individus de $Pop_{globale}$ avec $\boldsymbol{Pop_{globale} = Pop_t \cup Pop_{Cross} \cup Pop_{Mut}}$
27: $t = t + n_f$
28: **Fin Si**
29: **Si** $t > t_{max}$ alors
30: **Break ;**
31: **Fin Si**
32: Mise-à-jour (renouveler) g^*
33: $t = t + 1$
34: **Fin Tanque**



.

Il est très clair, d'après l'algorithme 4.1 que la structure principale de l'algorithme FPA est préservée, on observe aussi que l'algorithme FPA-GA est doté d'une partie supplémentaire contient l'application (l'intervention) de trois opérateurs de GAs(ou plutôt deux), puisque la sélection élite (Elite Sélection) est un mécanisme général que l'on peut trouver dans un grand nombre d'algorithmes.

Il à noter aussi que, l'intervention de la partie GAs est contrôlé par la condition suivante ;

$$\text{Nombre aléatoire} \in [0,1] < Prob\_Cr\_M \tag{4.1}$$

Dans cette dernière, le paramètre $Prob\_Cr\_Mu$ est appelé facteur de probabilité d'application des opérateurs GAs.Par conséquent, si la condition (4.1) est vérifiée, deux autres populations supplémentaires dites respectivement Pop_{Cross} et Pop_{Mut} seront ajoutées à travers une génération réalisée par la partie GAs. De ce fait, nous avons en somme trois populations différentes ; les deux relatives aux opérateurs GAs plus la population primitive Pop_t. En réalité, la modification introduite permet d'améliorer significativement la caractéristique *diversification* de la l'algorithme global FPA-GA. Aussi, comme nous le verrons par la suite à travers les résultats obtenus que cette dernière a aussi un impacte positif sur la précision de FPA-GA.

L'opération de génération d'une population globale dite $Pop_{Globale}$(voir l'équation 4.2), suivie par un mécanisme de sélection de bons vecteurs de solutions dit ; sélection élite.

$$Pop_{Globale} = Pop_t \cup Pop_{Cross} \cup Pop_{Mut} \tag{4.2}$$

Le résultat direct de l'application de cette sélection est l'obtention de $n_f$meilleurs vecteurs de solutions parmi les $3 \times n_f$ vecteurs relatifs aux trois populations générées. Ainsi, une conséquence rationnelle de cette tâche est le renforcement de la caractéristique *intensification* de l'algorithme inventé.

De plus, nous notons que l'algorithme FPA-GA contient (comme il est montré-voir Algorithme (4.1) deux structures de contrôle indépendantes qui forment les lignes 20jusqu'à 22 et les lignes29 jusqu'à31, dont leur but est d'éviter des exécutions supplémentaires d'évaluations de la fonction objective. Il à noter aussi que, le nombre d'évaluations de la fonction objective est toujours incrémenté tant que la condition suivante est vérifiée (voir la ligne 18 et27).

En effet, le point fort de l'algorithme FPA-GA proposé, résidé dans le fait qu'il a une structure algorithmique très simple par rapport à d'autres algorithmes, ce qui facilite énormément sa conception et son implémentation, même son temps de calcul !

La figure suivante nous montre aussi une représentation graphique d'une seule itération ($t = 1$) dansl'algorithme FPA-GA ;

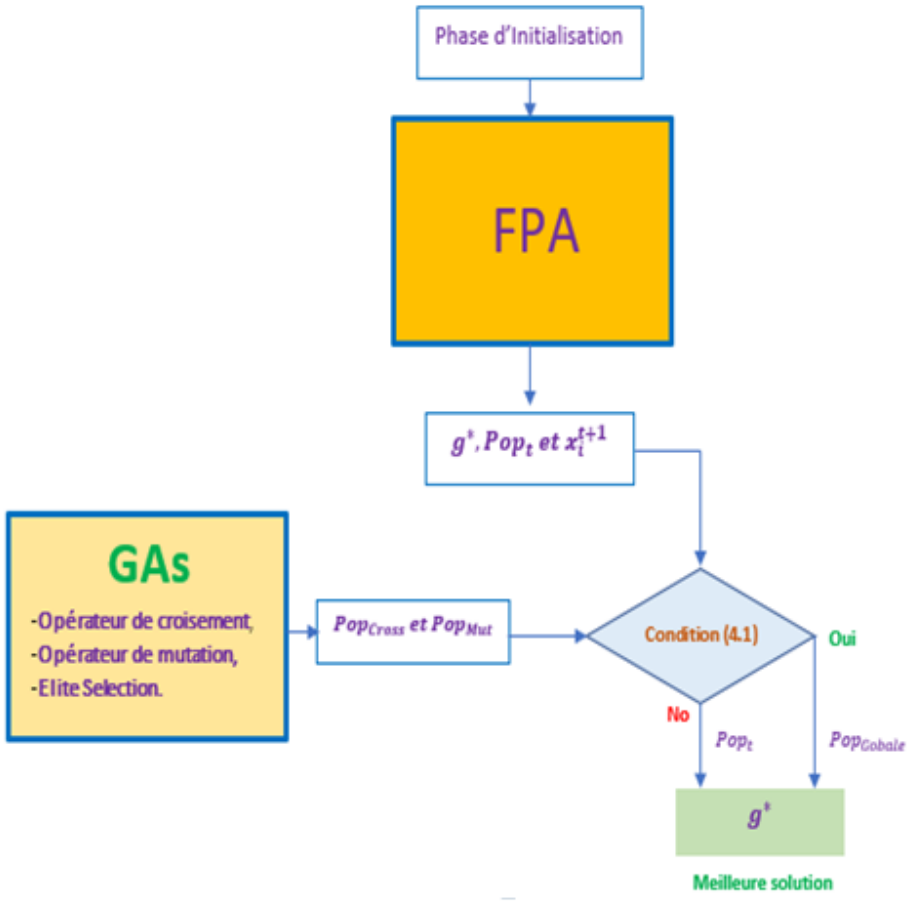


Figure 4.1 : Représentation graphique d'une seule itération dans l'algorithme FPA-GA développé.

Généralement, la fonction objectif représente un paramètre très important pour la réussite d'un processus ou une opération d'optimisation. En effet, son choix a un impact décisif sur la qualité des résultats voulus. Concernant notre variante FPA développé nous notons que nous évaluions la qualité des solutions à travers un ensemble de fonctions (objectifs) dites ; *Fonctions benchmarks* de différentes classes.

Aussi, il est à noter que les GAs sont largement utilisés en position hybride avec d'autres méthodes ou algorithmes d'optimisation. La référence suivante [55], qui est publiée sous le titre 'Hybridization in Genetic Algorithms' permet au lecteur de prendre une bonne idée sur ce sujet, malgré que cette dernière, est relativement récente.

4.3 Conclusion :

Dans ce chapitre, nous avons présenté en détail le principe de la nouvelle variante d'algorithme FPA développée. Cette dernière qui porte le nom FPA-GA, est le résultat d'une hybridation directe entre la version classique de l'algorithme FPA et deux opérateurs appartenant aux algorithmes génétiques GAs à savoir ; l'opérateur de croisement et celui de mutation suivis tous les deux par une opération de sélection dite (en anglais) ; Elite Sélection. Comme nous avons vu, l'algorithme développé a une structure algorithmique très simple en termes de conception et d'implémentation (programmation). En effet, ce dernier est le résultat d'une intégration directe des trois mécanismes (croisement, mutation et sélections élite) dans le corp algorithmique de l'algorithme FPA (sous sa forme standard). Aussi, comme nous le verrons par la suite dans le chapitre suivant (à travers les tests effectués), cette innovation permet d'améliorer significativement les capacités de l'algorithme FPA en termes de ; caractéristiques *diversification* et *intensification, et* précision.

Chapitre 5

CHAPITRE 5

Validation de la nouvelle variante de FPA ; le FPA-GA

5.1 Introduction :

Dans le présent chapitre, nous allons exposer en détaille la validation de notre variante FPA proposée. Plus précisément, afin d'argumenter les modifications introduites, nous travaillons à tester les capacités de l'algorithme FPA-GA préposé en utilisant un ensemble de fonctions de tests dites ; *fonctions benchmarks*. En réalité, ces fonctions sont définies dans le cadre de la conférence 2005 *IEEE Congress on Evolutionary Computation* (CEC 2005) [56]. Il est à noter que, cette méthode de validation considérée relativement traditionnelle (par effet d'apparition d'un grand nombre d'autres techniques) est très utilisée par l'ensemble des chercheurs travaillant dans ce domaine. Aussi, cette dernière a une crédibilité très acceptable en termes de ses capacités de montrer les puissances d'un tel algorithme vis-à-vis les défis imposés par les fonctions (dites aussi ; protocoles de tests) appartenant à la méthode.

Ce chapitre porte aussi une étude comparative entre le FPA-GA et d'autres algorithmes appartenant à la même famille (algorithmes évolutionnaires) afin de monter les capaciter de la méthode proposée.

5.2 Outils-tests-fonctions benchmarks CEC 2005 :

Afin d'évaluer la pertinence de la nouvelle variante hybride proposée, nous l'avons comparée comme nous l'avons dit précédemment à un ensemble d'algorithmes de même genre, existant dans la littérature (compris aussi la version FPA standard) sur un ensemble de fonctions-tests dites ; fonctions benchmarks CEC 2005.

Au sens simple, ces fonctions ont été proposées dans le cadre de la conférence 2005 *IEEE Congress on Evolutionary Computation* (CEC 2005) [56] afin de tester les capacités des différentes méthodes d'optimisation.

Ces fonctions sont groupées principalement en deux grandes catégories ; la première contient cinq fonctions dites ; *Fonctions Unimodales* (*Unimodal Functions*) constitué de F_1à F_5alors que la seconde a vingt fonctions dites ; Fonctions Multimodales (*Multi-modal Functions*) composé de

F_6à F_{25}. Plus précisément, les fonctions F_1à F_5 ne possèdent qu'un minimum global tandis que le second groupe divisé comme suit ; fonctions multimodales de base (*Single Function*) F_6 à F_{12}, fonctions étendues (*ExpandedFunction*) F_{13} à F_{14} et fonctions composées hybrides (*Hybrid Composition Function*) F_{15} à F_{25} a des fonctions hautement multimodales, i.e. le nombre de pics et de vallées augmente avec l'augmentation du nombre de dimensions. De plus, les fonctions F_{13} à F_{14} sont des versions biaisées et tournées des fonctions de base alors que F_{15} à F_{25} sont des fonctions composées c.-à-d. crées par hybridation de deux ou plusieurs fonctions de test.

Les différentes caractéristiques de ces fonctions ainsi que leurs expressions peuvent être vues dans ce qui suit [56] ; Il à noter que, nous avons donné les nomes de ces dernières en anglais afin d'éviter une dispersion de lecteur, puisque toutes les références utilisant ces fonctions donnent leurs noms en anglais même celles appartenant à la bibliographie francophone.

Des informations supplémentaires importantes sur toutes ces fonctions sont disponibles dans [56].

5.2.1 Fonctions Unimodales (Unimodal functions)

1. *Shifted Sphere function*

Selon [56], nous avons :

$$F_1(x) = \sum_{i=1}^{D} z_i^2 + f\_bias_1 , z = x - o, x = [x_1, x_2, \dots, x_D]$$

D:dimensions du problème, $\boldsymbol{o} = [o_1, o_2, \dots, o_D]$: l'optimum global décalé.
Avec $\boldsymbol{x} \in [-100,100]$, l'optimum global $\boldsymbol{x}^* = \boldsymbol{o}$, $F_1(\boldsymbol{x}^*) = f\_bias_1 = -450$.

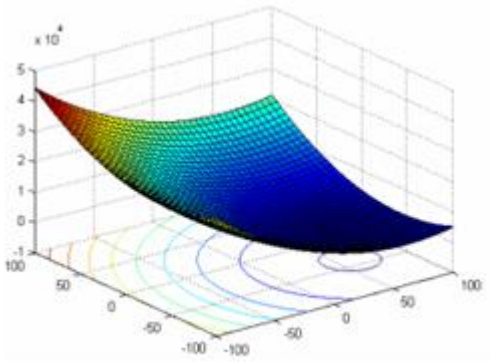


Figure 5.1 : Représentation 3-dimesnions pour la fonction F_1 (cas D=2) [56].

2. *Shifted Schwefel's problem 1.2*

Selon [56], nous avons :

$$F_2(x) = \sum_{i=1}^{D} \sum_{j=1}^{i} z_j + f\_bias_2 \,, z = x - o, x = [x_1, x_2, \ldots, x_D]$$

Ici, D:dimension du problème,$o = [o_1, o_2, \ldots, o_D]$: l'optimum global décalé.

Avec $x \in$ [-100,100]^D,l'optimum global : $x^* = o, F_2(x^*) = f\_bias_2 = -450$.

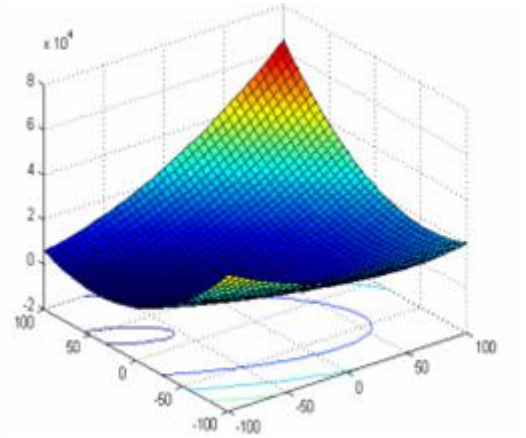


Figure 5.2 : Représentation 3-dimesnions pour la fonction F_2 (cas D=2) [56].

3. *Shifted rotated high conditioned elliptic function*

Selon [56], nous avons :

$$F_3(x) = \sum_{i=1}^{D} (10^6)^{\frac{i-1}{D-1}} z_i^2 + f\_bias_3 \,, z = (x - o) \times M, x = [x_1, x_2, \ldots, x_D]$$

Ici, D:dimension du problème,$o = [o_1, o_2, \ldots, o_D]$: l'optimum global décalé.

M : est une matrice orthogonale.

Avec $x \in$ [-100,100]^D,l'optimum global : $x^* = o, F_3(x^*) = f\_bias_3 = -450$.

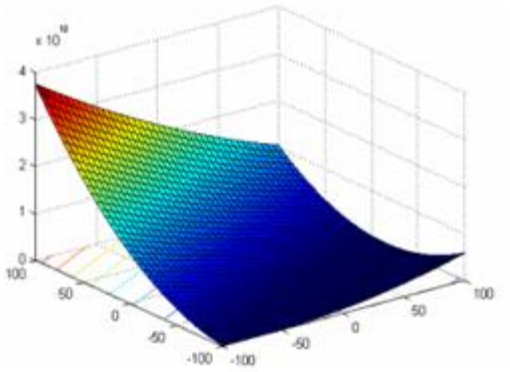


Figure 5.3 : Représentation 3-dimesnions pour la fonction F_3 (cas D=2) [56].

4. *Shifted Schwefel's problem 1.2 with noise in fitness*

Selon [56], nous avons :

$$F_4(x) = \left(\sum_{i=1}^{D}(\sum_{J=1}^{\mathrm{i}} \mathrm{z_i})^2\right) \times (1 + 0{,}4\,|\mathrm{N}(0,1)|) + f\_bias_4\,, z = x - o, x = [x_1, x_2, \ldots., x_D]$$

Ici, D: dimension du problème,$o = [o_1, o_2, \ldots., o_D]$: l'optimum global décalé.

M : est une matrice Orthogonale.

Avec $x \in$ [-100,100]^D,l'optimum global : $x^* = o, F_4(x^*) = f\_bias_4 = -450$.

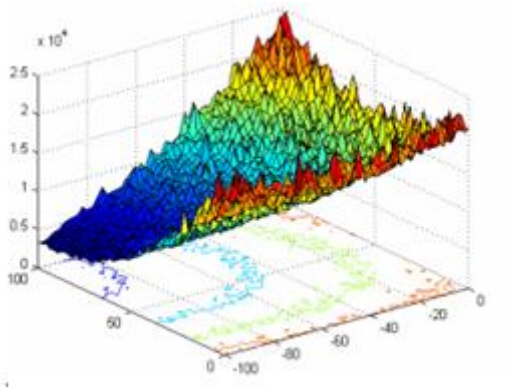


Figure 5.4 : Représentation 3-dimesnions pour la fonction F_4 (cas D=2) [56].

5. *Schwefel's problem 2.6 with global optimum on bounds*

Nous avons, $f(x) = max\{|x_1 + 2x_2 - 7|, |2\,x_1 + x_2 - 5|\}, i = 1, \ldots. n\,, x^* = [1,3]$, et $f(x^*) = 0$, étendre à une dimension D ; Par la suite, la fonction $F_5(x)$ est définie par ;

$$F_5(x) = max\{|A_i x - B_i|\} + f\_bias_5\,, i = 1, \ldots., D, x = [x_1, x_2, \ldots., x_D].$$

Ici, D: dimension du problème.

Aussi, Areprésente une matrice de dimension$D \times D$, avec a_{ij} (ses éléments) sont des nombres entiers aléatoire$\in [-500,500]$, $det(A) \neq 0$ avec A_i est le $i^{ième}$ rang dans$A$. $B_i = A_i \times O$, o est un vecteur de longueur D et o_i sont des nombres aléatoires dans l'intervalle$[-100,100]$.

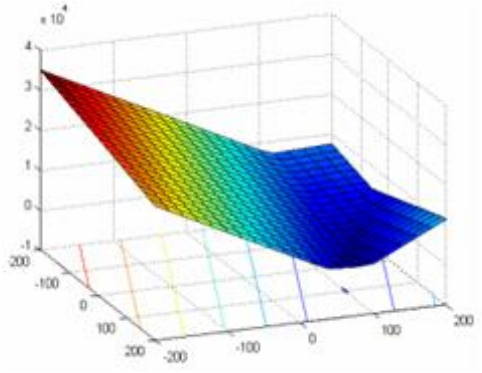


Figure 5.5 : Représentation 3-dimesnions pour la fonction F_5 (cas D=2) [56].

5.2.2 Fonctions multimodal's de base (Basic Multimodal Functions)

1. ***Shifted Rosenbrock's function***

Selon, nous avons :

$$F_6(x) = \sum_{i=1}^{D-1}(100(z_i^2 - z_{i+2})^2 + (z_i - 1)^2) + f\_bias_6, z = x - o, x = [x_1, x_2, \dots, x_D]$$

Ici, D: dimension du problème, $o = [o_1, o_2, \dots, o_D]$: l'optimum global décalé.

Avec, $x \in [-100,100]^D$, l'optimum global : $x^* = o, F_6(x^*) = f\_bias_6 = 390$.

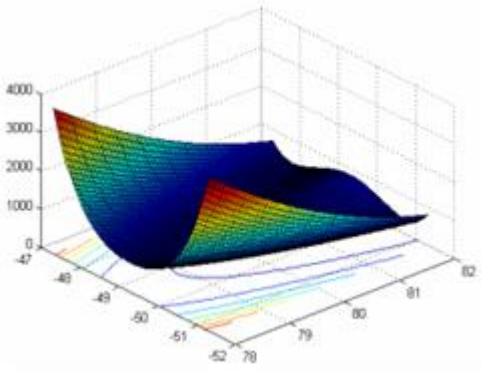


Figure 5.6 : Représentation 3-dimesnions pour la fonction F_6 (cas D=2) [56].

2. ***Shifted rotated Griewank's function without bounds:***

Nous avons,

$$F_7(x) = \sum_{i=1}^{D} \frac{Z_i^2}{4000} - \prod_{i=1}^{D} cos\left(\frac{z_i}{\sqrt{i}}\right) + 1 + f\_bias_7, z = (x - o) \times M, x = [x_1, x_2, \dots, x_D]$$

Ici, D: dimension du problème, $o = [o_1, o_2, \dots, o_D]$: l'optimum global décalé.

M' Représente la transformation linéaire de matrice, condition numéro 3, avec $M = M'(1 + 0.3|N(0,1)|)$.

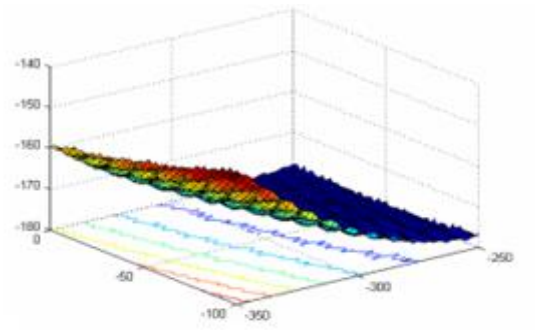


Figure 5.7 : Représentation 3-dimesnions pour la fonction F_7 (cas D=2) [56].

3. Shifted rotated Ackley's function with global optimum on bounds:

Nous avons,

$F_8(x) = -20\exp(-0{,}2\sqrt{\frac{1}{D}\sum_D^1 z_i^2}) - \exp(\frac{1}{D}\sum_{l=1}^{D}\cos(2\pi z_i)) + 20 + e + f\_bias_8$,

$z = (x - o) \times M, x = [x_1, x_2, \ldots., x_D]$.

Ici, D: dimension du problème,$o = [o_1, o_2, \ldots., o_D]$: l'optimum global décalé.

M' Représente la transformation linéaire de matrice$M$, condition numéro 100.

Avec $x \in$ [-32,32]^D,l'optimum global : $x^* = o, F_8(x^*) = f\_bias_6 = -140$.

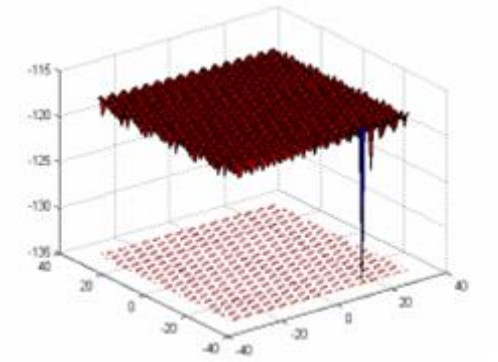


Figure 5.8 : Représentation 3-dimesnions pour la fonction F_8 (cas D=2) [56].

4. ShiftedRastrigin'sfunction

Nous avons,

$$F_9(x) = \sum_{i=1}^{D}(z_i^2 - 10cos(2\pi z_i) + 10) + f\_bias_9, z = x - o, x = [x_1, x_2, \dots, x_D]$$

Ici, D: dimension du problème, $o = [o_1, o_2, \dots, o_D]$: l'optimum global décalé.

Avec $x \in [-5,5]^D$, l'optimum global : $x^* = o$, $F_9(x^*) = f\_bias_9 = -330$.

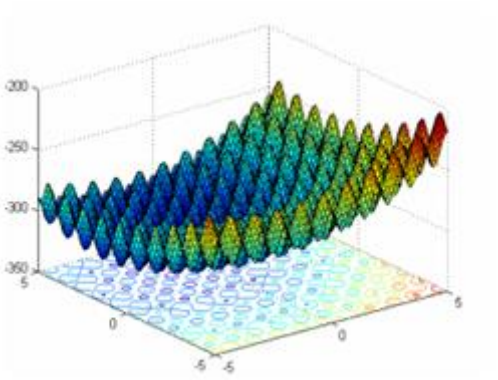


Figure 5.9 : Représentation 3-dimesnions pour la fonction F_9 (cas D=2) [56].

5. Shifted rotated Rastrigin's function

Nous avons,

$$F_{10}(x) = \sum_{i=1}^{D}(z_i^2 - 10\cos(2\pi z_i) + 10) + f\_bias_{10}, z = (x - o) \times M, x = [x_1, x_2, \dots, x_D]$$

Ici, D: dimension du problème, $o = [o_1, o_2, \dots, o_D]$: l'optimum global décalé.

M' représente la transformation linéaire de matrice M, condition numéro 2.

Avec $x \in [-5,5]^D$, l'optimum global : $x^* = o$, $F_{10}(x^*) = f\_bias_{10} = -330$.

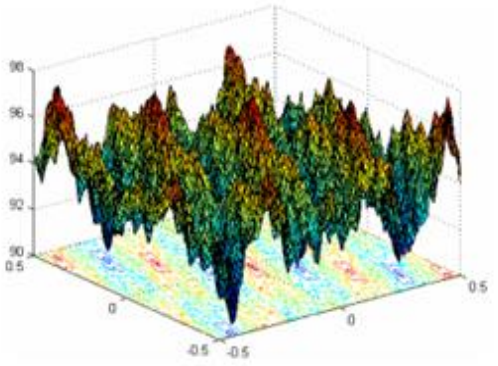


Figure 5.10 : Représentation 3-dimesnions pour la fonction F_{10} (cas D=2) [56].

6. Shifted rotated Wrierstrass function

Nous avons

$$F_{11}(x) = \sum_{i=1}^{D}(\sum_{k=0}^{kmax}\left[a^k cos\left(2\pi b^k(z_i + 0.5)\right)\right] - D\sum_{k=0}^{kmax}\left[a^k cos\left(\pi b^k\right)\right]) + f\_bias_{11},$$

Avec $a = 0.5$, $b = 3$ et $k_{max} = 20$, $z = (x - o) \times M$, $x = [x_1, x_2, \dots, x_D]$.

Ici, D: dimension du problème,$o = [o_1, o_2, \dots, o_D]$: l'optimum global décalé.

Avec $x \in$ [-0.5, 0. 5]^D,l'optimum global : $x^* = o, F_{11}(x^*) = f\_bias_{11} = 90$.

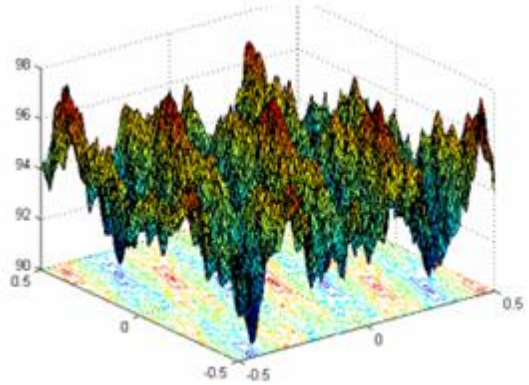


Figure 5.11 : Représentation 3-dimesnions pour la fonction F_{10} (cas D=2) [56].

7. Schwefel's problem 2.13

Nous avons,

$$F_{12}(x) = \sum_{i=1}^{D}[A_i - B_i(x)]^2 + f\_bias_{12}, x = [x_1, x_2, \dots, x_D]$$

$A_i = \sum_{j=1}^{D}\left[a_{ij}sin\alpha_j + b_{ij}cos\alpha_j\right]^2$, $B_i(x) = \sum_{j=1}^{D}\left[a_{ij}sinx_j + b_{ij}cosx_j\right]^2$ pour$i = 1, .., D$.

Ici, D:Dimension du problème,

Aussi, A et B représente deux matrices de dimension $D \times D$, avec a_{ij} et b_{ij} (leurs éléments) sont des nombres entiers aléatoire ∈ $[-100,100]$, $\alpha = [\alpha_1, \alpha_2, \dots, \alpha_D]$ avec α_j sont des nombres aléatoires dans $[-\pi, +\pi]$ .Avec aussi,$x \in [-\pi, +\pi]$,l'optimum global : $x^* = \alpha$, $F_{11}(x^*) = f\_bias_{12} = -460$.

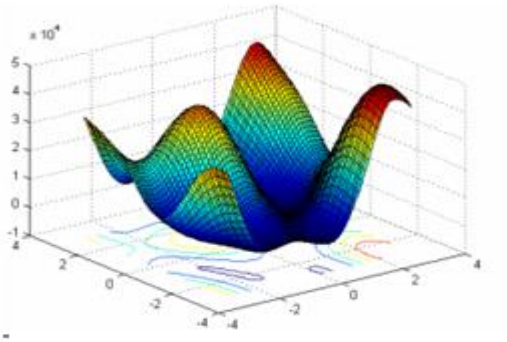


Figure 5.12 : Représentation 3-dimesnions pour la fonction F_{12} (cas D=2) [56].

5.2.3 Fonctions étendues (Expanded Functions)

Selon [56], en utilisant une fonction de$2 \times D$dite $F\ (x, y)$comme une fonction de départ, la fonction étendue correspondante et exprimée comme suit ;

$$EF\ (x_1, x_2, .., x_D) = F(x_1, x_2) + \ F(x_2, x_3) + F(x_{D-1}, x_D) + F(x_D, x_1)$$

En somme, nous avons deux fonctions de ce genre F_{13} et F_{14} donnée comme suit ;

1. Shifted expanded Griewank's plus Rosenbrock's function (F_8F_2)

Selon [56], nous avons ;

$F_8(x)$: Griewank's fonction : $F_8(x)$=$\frac{x_i^2}{4000}$−$\prod_{i=1}^{D} \cos(\frac{x_i}{\sqrt{i}})\ + 1$,

$F_2(x)$: Rosenbrock's fonction : $F_2(x)$=$\sum_{i=1}^{D-1}(100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2)$,

Nous avons alors selon la définition précédente ;

$F_8F_2(x_1, x_2, .., x_D) = F_8\big(F_2(x_1, x_2)\big) + F_8\big(F_2(x_2, x_3)\big) + \cdots + F_8\big(F_2(x_{D-1}, x_D)\big) + F_8\big(F_2(x_D, x_1)\big)$

Décalée à ($shift\ to$-en anglais) à une fonction $F_{13}(x)$ comme suit;

$F_{13}(x) = F_8F_2(x_1, x_2, .., x_D) + f\_bias_{13}$, avec, $z = x - o + 1$, x =$[x_1, x_2, \ldots., x_D]$

Ici, D: Dimension du problème,$o = [o_1, o_2, \ldots., o_D]$: l'optimum global décalé.

Avec aussi, $x \in$ [-5, +5]^D,l'optimum global : $x^* = o$, $F_{13}(x^*) = f\_bias_{13} = -130$.

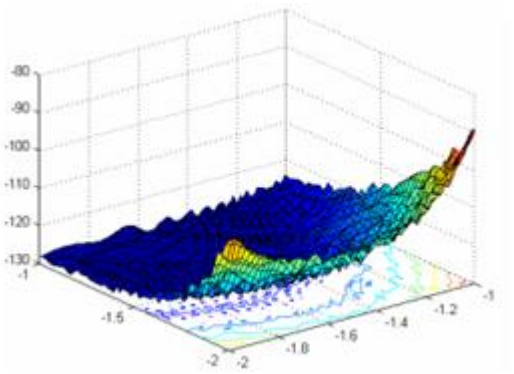


Figure 5.13 : Représentation 3-dimesnions pour la fonction F_{13} (cas D=2) [56].

2. Shifted rotated expanded Scaffer's F_6 function

Selon [56], nous avons;

$$F(x, y) = 0.5 + \frac{(sin^2(\sqrt{x^2+y^2} - 0{,}5)}{(1+0.001(x^2+y^2))^2}$$

Étendue à ;

$$F_{14}(x) = EF\ (z_1, z_2, .., z_D) = F(z_1, z_2) + F(z_2, z_3) + \cdots + F(z_{D-1}, z_D) + F(z_D, z_1) + f\_bias_{14}$$

$z = (x - o) \times M$, x =$[x_1, x_2, \ldots., x_D]$

Ici, D: dimension du problème,$o = [o_1, o_2, \dots, o_D]$: l'optimum global décalé.

M' Représente la transformation linéaire de la matrice M sous la condition numéro 3.

Avec aussi, $x \in$ [-100, +100]^D,l'optimum global : $x^* = o$, $F_{14}(x^*) = f\_bias_{14} = -300$.

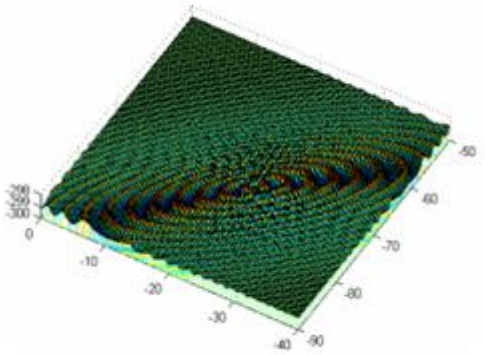


Figure 5.14 : Représentation 3-dimesnions pour la fonction F_{14} (cas D=2) [56].

5.2.4 Fonctions composées (Composition functions)

Selon [56], nous avons les constatations suivantes ;

$F(x)$: Nouvelle fonction composée,

$f_i(x)$= i^{th}fonction de base utilisée pour construire la fonction composée,

n : Nombre de fonction de base,

D : Dimensions,

M_i : Matrice de transformation linéaire pour chaque$f_i(x)$,

o_i **:** Nouvelle position optimale décalée pour chaque$f_i(x)$.

$$F(x) = \sum_{i=1}^{n}\{w_i \times [f_i'((\mathrm{x} - o_i)/\lambda_i \times M_i) + bias_i]\} + f\_bias_{15}$$

Ici, w_i représente le poids pour chaque$f_i(x)$, calculé comme suit :

$$w_i = exp\left(-\frac{\sum_{k=1}^{D}(x_k - o_i)^2}{2D\sigma_i^2}\right)$$

Avec ;

$$w_i = \begin{cases} w_i & w_i = \max(w_i) \\ w_i \times (1 - \max(w_i)^{10}) & w_i \neq \max(w_i) \end{cases}$$

Plus une normalisation des poids selon ;

$$w_i = w_i / \sum_{i=1}^{n} w_i$$

Avec ;

σ_i : est un paramètre utilisé pour contrôler la plage de couverture relative à la fonction$f_i(x)$,

λ_i : est un paramètre utilisé compresser la fonction ; selon [56] nous avons deux cas relatifs à ce dernier ;$\lambda_i >$ signifié étendue et$\lambda_i < 1$ signifié une compression,

o_i : Défini les optimaux globaux et locaux,

f_bias : défini quel optimum est un optimum global.

Il est à noter que, en utilisant o_i etf_bias un optimum global peut être placé n'importe où.

Également, si $f_i(x)$ sont des fonctions différentes, ceci implique que différentes fonctions ont des propriétés et des hauteurs différentes. En effet, afin d'obtenir une meilleure fusion, uneestimation ded'une plus grande valeur dite$f_{\max i}$est faite pour 10 fonctions $f_i(x)$, puis unenormalise de chaque fonction de base à des hauteurs similaires est réalisée comme suit ;

$$f_i'(x) = C \times f_i(x)/|f_{\max i}|$$

Sachant que, $|f_{\max i}|$ est estimée comme suit ;

$$|f_{\max i}| = f_i\left(\left(\frac{x'}{\lambda_i}\right) \times M_i\right) \text{ Avec :} x' = [5,5\ldots,5].$$

Dans les expressions des fonctions composées qui sera exposées, nous avons :

Nombre de fonctions de base$n = 10$,

o **:** matrice de dimension$n \times D$ définit les optimaux globaux relatifs aux fonctions $f_i(x)$.

$f\_bias_{15} = [0,100,200,300,400,500,600,700,800,900]$.

1. Hybrid composition function

Selon [56], nous avons ;

$f_{1-2}(x)$: Rastrigin's Function ; $f_i(x)$=$\sum_{i=1}^{D}(x_i^2 - 10\cos(2\pi x_i) + 10)$,

$f_{3-4}(x)$: Wrierstrass Function;

$$f_i(x) = \boldsymbol{f_i(x)} = \left(\sum_{i=1}^{D}\left(\sum_{k=1}^{k\,max}[a^k \cos\left(2\pi b^k(x_i + 0.5)\right)]\right) - D\sum_{k=0}^{k\,max}[a^k \cos(2\pi b^k . 0.5)]\right),$$

Avec ; $a = 0.5$, $b = 3$, $k_{max} = 20$.

$f_{5-6}(x)$: Griewank's Function: $f_i(x)$=$\sum_{i=1}^{D}\frac{x_i^2}{4000} - \prod_{i=1}^{D}\cos(\frac{x_i}{\sqrt{i}}) + 1$

$f_{7-8}(x)$: Ackley's Function;

$$f_i(x) = -20\exp(-0{,}2\sqrt{\frac{1}{D}\sum_{i=1}^{D}x_i^2}) - \exp(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)) + 20 + e$$

$f_{9-10}(x)$: Sphere Function: $f_i(x) = \sum_{i=1}^{D} x_i^2$

On a aussi, $\sigma_i = 1$pour $i = 1,2,\ldots,D$ et $\lambda = [1, 1, 10, 10{,}5\ /60{,}5/60{,}5/32{,}5/32{,}5/100{,}5/100]$.

$\boldsymbol{M_i}$: Sont toutes des matrices d'identité.

Il est à noter que, ces formules ne concernent que les fonctions de base, aucun changement ou rotation n'est inclus dans ces expressions, $x$ici représente tout simplement une variable dans une fonction. Prendre f_1 comme un exemple, quand on calcule$f_1'((x - o_1)/\lambda_1 \times M_1)$ il faut par la suite calculer $f_1(z)$ comme suit ;

$$f_1(z) = \sum_{i=1}^{D}(z_i^2 - 10\cos(2\pi z_i) + 10) \text{ avec} z = ((x - o_1)/\lambda_1) \times M_1$$

Pour la fonction considérée, $x \in [-5, +5]^D$,l'optimum global : $\boldsymbol{x}^* = \boldsymbol{o}$, $F_{15}(\boldsymbol{x}^*) = f\_bias_{15} = 120$.

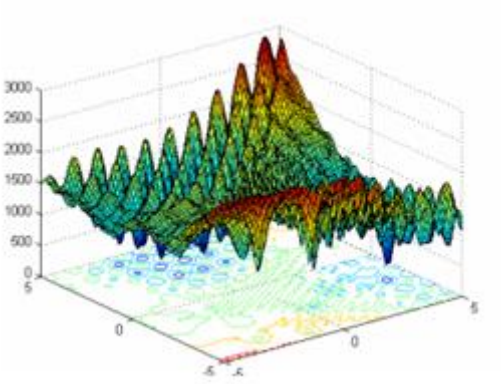


Figure 5.15 : Représentation 3-dimesnions pour la fonction F_{15} (cas D=2) [56].

2. Rotated version of hybrid composition function F_{16}

Dans ce cas, sauf que $M_i$sont différentes matrices de transformation linéaire avec un nombre de condition 2, toutes les autres les règles sont les même que le cas de la fonctionF_{15}. Pour plus d'information sur cette fonction, veuillez consulter la référence [56].

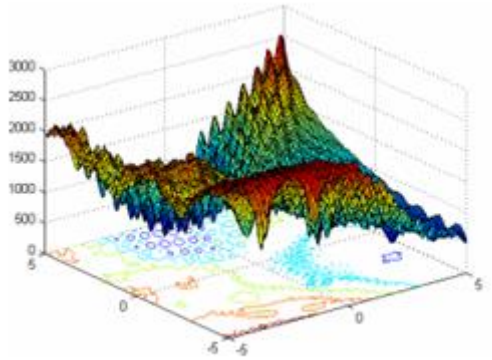


Figure 5.16 : Représentation 3-dimesnions pour la fonction F_{16} (cas D=2) [56].

3. F_{16} with noise in fitness

Nous avons [56], $G(x) = (F_{16})$ alors :

$$F_{17}(x) = G(x)^*(1 + 0{,}2\,|N(0{,}1)|) + f\_bias_{17}$$

Tous les paramètres sont les mêmes que F_{16}.

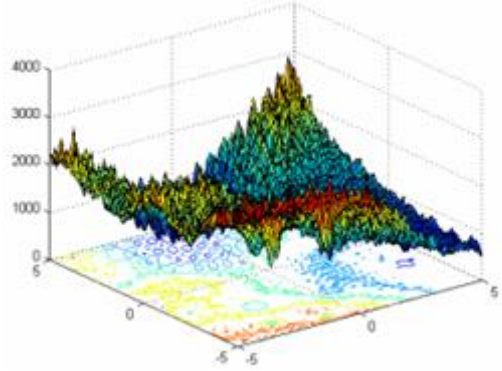


Figure 5.17 : Représentation 3-dimesnions pour la fonction F_{17} (cas D=2) [56].

3. Rotated Hybrid Composition Function

Selon [56], nous avons;

$f_{1-2}(x)$: Ackley's Function;

$$f_i(x) = -20\exp(-0{,}2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}) - \exp(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)) + 20 + e$$

$f_{3-4}(x)$: Rastrigin's Function: $f_i(x)=\sum_{i=1}^{D}(x_i^2 - 10\cos(2\pi x_i) + 10)$,

$f_{6-8}(x)$: Sphere Function:$f_i(x) = \sum_{i=1}^{D} x_i^2$

$f_{9-10}(x)$: Wrierstrass Function :

$$f_i(x) = \boldsymbol{f_i(x)} = \left(\sum_{i=1}^{D}\left(\sum_{k=1}^{k\,max}\left[a^k \cos\left(2\pi b^k(x_i + 0.5)\right)\right]\right) - D\sum_{k=0}^{k\,max}\left[a^k \cos(2\pi b^k . 0.5)\right]\right),$$

Avec ;$a = 0.5, b = 3, k_{max} = 20$.

$f_{11-12}(x)$: Griewank's Function:

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

On a aussi,$\sigma = [1,2,1.5,1.5,1,1,1.5,1.5,2,2]$;pour et $\lambda = [10/32\ ;\ 5/32\ ;\ 2\ ;\ 1\ ;\ 10/100\ ;\ 5/100\ ;\ 20\ ;\ 10\ ;\ 10/60\ ;\ 5/60]$.

M_i : Sont toutes des matrices de rotation, les numéros de conditions sont respectivement [2,3,2,3 2,3,20,30,200,300], $o_{10} = [0,0,\ldots,0]$.

Aussi, pour plus d'informations sur cette fonction le lecteur peut consulter la référence [56].

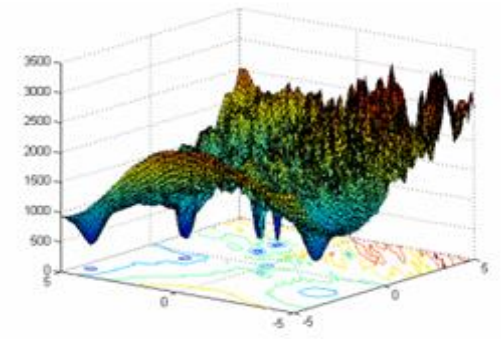


Figure 5.18 : Représentation 3-dimesnions pour la fonction F_{18} (cas D=2) [56].

3. Rotated Hybrid Composition Function with narrow basin global optimum

A propos de cette fonction, tous ses paramètres sont les mêmes que F_{18} sauf les exceptions suivantes ;

$\sigma = [0.1,1,2,1.5,1.5,1,1,1.5,1.5,2,2]$; pour et $\lambda = [0.5/32\ ;\ 5/32\ ;\ 2\ ;\ 1\ ;\ 10/100\ ;\ 5/100\ ;\ 20\ ;\ 10\ ;\ 10/60\ ;\ 5/60]$.

Aussi, pour plus d'informations sur cette fonction le lecteur peut consulter la référence [56].

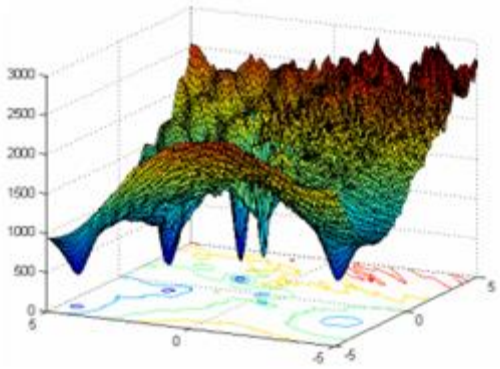


Figure 5.19 : Représentation 3-dimesnions pour la fonction F_{19} (cas D=2) [56].

4. Rotated Hybrid Composition Function with Global Optimum on the Bounds

Même chose pour cette fonction, tous ses paramètres sont les mêmes que F_{19} sauf les exceptions suivantes ; $\sigma_{1(2j)} = 5$ pour $j = 1,2, \ldots \ldots, \frac{D}{2}$ .Plusieurs informations sur cette fonction sont disponibles dans [56].

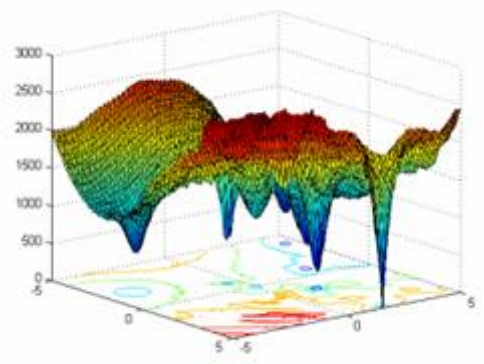


Figure 5.20 : Représentation 3-dimesnions pour la fonction F_{20} (cas D=2) [56].

5. Rotated Hybrid Composition Function

Selon [56], nous avons;

$f_{1-2}(x)$: Rotated Expanded Scaffer's F_6 Function;

$$F(x,y) = 0.5 + \frac{(sin^2(\sqrt{x^2+y^2}-0,5)}{(1+0.0001(x^2+y^2))^2}$$

$$f_i(x) = F\,(x_1, x_2\,) + F\,(x_2, x_3) + \cdots + F(x_{D-1}, x_D) + F(x_D, x_1)$$

$f_{1-2}(x)$: Rastrigin's Function : $f_i(x) = \sum_{\boldsymbol{i=1}}^{D}(x_i^2 - 10\cos(2\pi x_i) + 10)$

$f_{2-3}(x)$ **:** F_8F_2 Function:

$$F_8(x) = \sum_{i=1}^{\boldsymbol{D}} \frac{x_i^2}{4000} - \prod_{i=1}^{D} cos(\frac{x_i}{\sqrt{i}}) + 1$$

$$F_2(x) = \sum_{i=1}^{D-1}(100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2)$$

$$f_i(x) = F_8\left(F_2(x_1, x_2\,)\right) + F_8\left(F_2(x_2, x_3\,)\right) + \cdots + F_8\left(F_2(x_{D-1}, x_D\,)\right) + F_8\left(F_2(x_D, x_1\,)\right)$$

$f_{4-5}(x)$ **:** Wrierstrass Function:

$$f_i(x) = \boldsymbol{f_i}\mathbf{(x)} = (\sum_{i=1}^{D}(\sum_{k=1}^{\mathrm{k\,max}}[\mathrm{a^k}\cos\left(2\pi \mathrm{b^k}(\mathrm{x_i} + 0.5)\right)]) - \mathrm{D}\sum_{\mathrm{k=0}}^{\mathrm{k\,max}}[\mathrm{a^k}\cos(2\pi \mathrm{b^k}.\,0.5)]\,,$$

Avec ;$a = 0.5, b = 3, k_{max} = 20$.

$f_{6-7}(x)$ **:** Griewank's Function :

$$f_i(x) = \sum_{i=1}^{\boldsymbol{D}} \frac{x_i^2}{4000} - \prod_{i=1}^{D} \cos(\frac{x_i}{\sqrt{i}}) + 1$$

$\sigma = [1,1,1,1,1,2,2,2,2,2]$,

$\lambda = [25/100\,;\ 5/100\,;\ 5\,;1\,;5;1\,;\ 50\,;\ 10\,;\ 25/200\,;5/200]$.

M_i : Sont des matrices orthogonales.

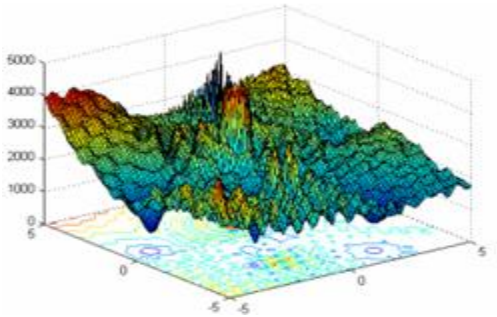


Figure 5.21 : Représentation 3-dimesnions pour la fonction F_{21} (cas D=2) [56].

6. Rotated Hybrid Composition Function with High Condition Number Matrix

Pour cette dernière, toutes les règles sont les même que F_{21} sauf les numéros des conditions relatifs aux M_i qui sont ; [10 20 50 100 200 1000 2000 3000 4000 5000].

Des informations supplémentaires sur cette fonction son disponible dans [56].

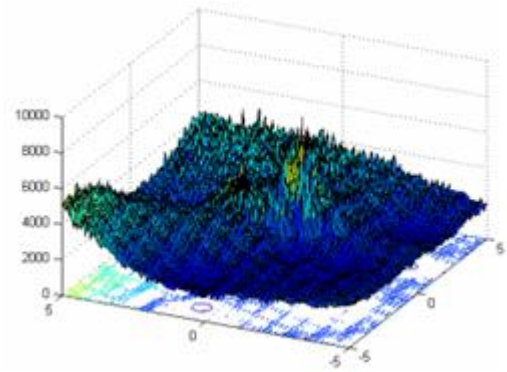


Figure 5.22 : Représentation 3-dimesnions pour la fonction F_{22} (cas D=2) [56].

7. Non-Continuous Rotated Hybrid Composition Function

Toutes les règles sont les même que F_{21}.Aussi, nous avons ;

Sauf x_j=$\begin{cases} x_j & |x_j - o_{1j}| < 1/2 \\ round(2x_j)/2 & |x_j - o_{1j}| >= 1/2 \end{cases}$ pour j=1,2,…, D

Rond (x)=$\begin{cases} a-1 & pour \quad x > 0 \; et \; b >= 0.5 \\ a & pour \quad b < 0.5 \\ a+1 & pour \quad x > 0 \; et \; b >= 0.5 \end{cases}$

Avec a est la partie réelle de x et $b$est la partie décimale de$x$.

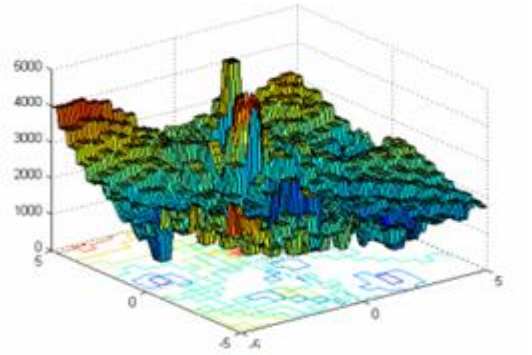


Figure 5.23 : Représentation 3-dimesnions pour la fonction F_{23} (cas D=2) [56].

8. Rotated Hybrid Composition Function :

Selon [56], nous avons ;

$f_1(x)$ **:** Wrierstrass Function:

$$f_i(x) = \boldsymbol{f_i}\mathbf{(x)}= (\sum_{i=1}^{D}(\sum_{k=1}^{k\,max}[a^k \cos\left(2\pi b^k(x_i + 0.5)\right)]) - D\sum_{k=0}^{k\,max}[a^k \cos(2\pi b^k.0.5)]\,,$$

Avec ;$a = 0.5, b = 3, k_{max} = 20$

$f_2(x)$: Rotated Expanded Scaffer's F_6 Function;

$$F(x,y) = 0.5 + \frac{(sin^2(\sqrt{x^2+y^2}-0{,}5)}{(1+0.0001(x^2+y^2))^2}$$

$$f_i(x) = F\,(x_1, x_2\,) + F\,(x_2, x_3) + \cdots + F(x_{D-1}, x_D) + F(x_D, x_1)$$

$f_3(x)$ **:** F_8F_2 Function

$$F_8(x) = \sum_{i=1}^{\boldsymbol{D}} \frac{x_i^2}{4000} - \prod_{i=1}^{D} cos(\frac{x_i}{\sqrt{i}}) + 1$$

$$F_2(x) = \sum_{i=1}^{D-1}(100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2)$$

$$f_i(x) = F_8\left(F_2(x_1, x_2\,)\right) + F_8\left(F_2(x_2, x_3\,)\right) + \cdots + F_8\left(F_2(x_{D-1}, x_D\,)\right) + F_8\left(F_2(x_D, x_1\,)\right)$$

$f_4(x)$: Ackley's Function

$$f_i(x) = -20\exp(-0{,}2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}) - \exp(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)) + 20 + e$$

$f_5(x)$: Rastrigin's Function: $f_i(x) = \sum_{i=1}^{D}(x_i^2 - 10\cos(2\pi x_i) + 10)$

$f_6(x)$: Griewank's Function: $f_i(x) = \sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{D} \cos(\frac{x_i}{\sqrt{i}}) + 1$

$f_7(x)$: Non-Continuous Expanded Scaffer's F6 Function

$$F(x,y) = 0.5 + \frac{(sin^2(\sqrt{x^2+y^2}-0{,}5)}{(1+0.0001(x^2+y^2))^2},$$

$$f(x) = F\,(y_1, y_2\,) + F\,(y_2, y_3) + \cdots + F(y_{D-1}, y_D) + F(y_D, y_1)$$

$$y_j = \begin{cases} x_j & |x_j| < 1/2 \\ round(2x_j)/2 & |x_j| >= 1/2 \end{cases} \quad \text{pour } j = 1,2,\ldots,D$$

$f_8(x)$: Non-Continuous Rastrigin's Function:$f_i(x) = \sum_{i=1}^{D}(y_i^2 - 10\cos(2\pi y_i) + 10)$,

$$y_j = \begin{cases} x_j & |x_j| < 1/2 \\ round(2x_j)/2 & |x_j| >= 1/2 \end{cases} \quad \text{pour} j = 1,2,\dots,D$$

$f_9(x)$: High Conditioned Elliptic Function: $f_i(x) = \sum_{i=1}^{D}(10^6)^{\frac{i-1}{D-1}} x_i^2$,

$f_{10}(x)$: Sphere Function with Noise in Fitness: $f_i(x) = (\sum_{i=1}^{D} x_i^2)\,(1 + 0{,}1\,|\mathrm{N}(0,1)|)$

Avec, σ_i =2, pour $i = 1,2 \dots, D$ et $\lambda =$ [10 ; 5/20 ; 1 ; 5/32 ; 1 ; 5/100 ; 5/50 ; 1 ; 5/100 ; 5/100]

M_i : Sont toutes des matrices de rotation, les numéros de condition

sont :[100 50 30 10 5 5 4 3 2 2].

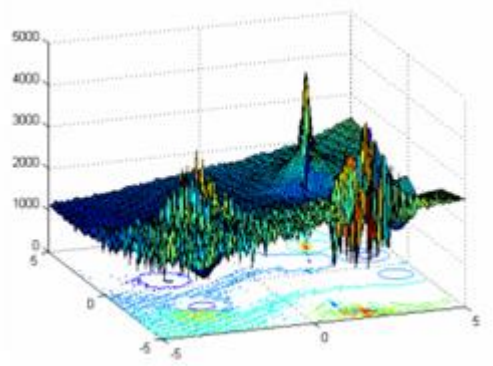


Figure 5.24 : Représentation 3-dimesnions pour la fonction F_{24} (cas D=2) [56].

9. Rotated Hybrid Composition Function without bounds

Toutes les règles sont les même que F_{24}, sauf que la plage de recherche exacte n'est pas définie.

5.3Algorithmes utilisés dans l'étude comparative :

Afin de prouver, l'efficacité de notre algorithme proposé, nous avons comparé tous les résultats obtenus par ce dernier avec ceux d'autres méthodes (algorithmes) appartenant à la même famille (algorithmes évolutionnaires). Plus précisément, nous avons utilisé les algorithmes suivants :

- La version standard de l'algorithme FPA,

- L'algorithme MGOFPA (Modified Generalised Opposition-based FPA) proposé par Amer Draa [57]. En effet, MGOFPA est une variante de l'algorithme FPA basée sur l'hybridation d'une technique d'apprentissage dite ; Generalised Opposition-based Learning avec un algorithme FPA modifié (Modified FPA),

- L'algorithme dit ; Covariance Matrix Adaptation Evolution Stratégies (CMAES) proposé par Nikolaus Hansen et Andreas Ostermeier [58]. Ce dernier, est un algorithme évolutionnaire basé sur une stratégie d'évolution permet d'adapter la matrice dite ; de covariance (CM- Covariance

Matrix) relative à la distribution de recherche intégrée dans le corps algorithmique de la méthode.

- L'algorithme dit ; Compréhensive Learning Particle Swarm Optimiser (CLPSO) proposé par Liang et ses collègues. Cet algorithme est une variante prometteuse de l'algorithme PSO (Particle Swarm Optimiser), il est basé sur l'introduction de quelques modifications de base sur la version primitive (l'algorithme PSO) (voir la référence [59] pour plus d'informations).

- Les trois variantes ; JADE, jDE et Codé de l'algorithme à évolution différentielle dit en anglais ; Différéntial Evolution (DE). Les noms de ces derniers sont les abréviations des algorithmes suivants ; JADE (Adaptive Differential Evolution with Optional External Archive) [60], jDE (publié sous le titre ; Self-Adapting Control Paramètres in Differential Evolution : A Comparative Study on Numerical Benchmark Problems) [61]et CoDE (publié sous le titre ; Differential évolution with composite trial vector génération stratégies and control paramètres) [62]. En effet, ces trois versions modifiées de DE, ont été proposées avec des modifications dont le but est d'améliorer le fonctionnement de l'algorithme original.

5.4Paramétrages d'algorithmes et méthodes de tests :

Nous présentons dans cette sous-section la liste des paramètres utilisés dans notre étude expérimentale en terme algorithmique (simulation). Les paramètres des algorithmes FPA-GA, MGOFPA et FPA sont donnés comme suit ;

-Paramètre de commutation $p = 0.2$ (le même pour le FPA et le MGOFPA),

-Nombre de population $n_f = 0.2$ (le même pour le FPA et le MGOFPA),

-Probabilité d'application de la méthode Modified Generalised Opposition-based Learning (MGO) dans le MGOFPA est égale à $Prob\_MGO = 0.1$ [57],

Concernant l'algorithme GA, nous avons ;

-Facteur de probabilité d'application des opérateurs GAs ; $Prob\_Cr\_Mu = 0.1$,

-Paramètre de croisement (crossover probabilité)$p_C = 0.55$,

-Taux de mutation est défini par la formule suivante ; $p_m = 1 - \frac{0.1}{N_b} \times i, i = 1, \dots\dots, N_b$, il est utilisé (dans le script de l'algorithme GA) sous la condition suivante ; $p_m < p_{mrand}$ avec p_{mrand} est un nombre aléatoire,

- Il est à noter que, nous avons utilisé les opérateurs de croisement et de mutation de types suivants ; croisement arithmétique (Arithmetical Crossover) et mutation non-uniforme (Non-Uniform mutation).

Concernant les algorithmes CMA-ES, CLPSO, JADE, jDE et CoDE le lecteur peut voir leur jeu de paramètres dans les références suivantes ; [58] pour le CMA-ES, [59] pour le CLPSO, [60] pour le JADE, [61] pour le jDE et [62] pour le CoDE.

Un test statistique de *Wilcoxon rank-sum* [voir la commande rank-sum dans le help du logiciel MATLAB], est appliqué sur les résultats de tous les algorithmes (FPA-GA, FPA, MGOFPA, CMA-ES, CLPSO, JADE, jDE et CoDE), avec un niveau de confiance égal à95%, afin de déterminer si leurs performances sont significativement différentes ou non.

Tous les algorithmes, ont été exécutés 25 fois (ou 25 essais) pour un nombre maximum de générations fixé à$\left(t_{max} = \left(n_f = N_p\right) \times 100000\right)$. Ainsi, les valeurs des meilleures solutions trouvées -à la fin de chaque essai-ont été utilisées pour juger l'existence d'une différance de qualité au sens statistique entre les algorithmes concourants. A cet effet, et afin de faciliter la conception nous avons utilisé les symboles suivants⊖, ⊕ et ⊙qui indiquent respectivement que ; le FPA-GA a donné de mauvais résultats par rapport à son algorithme rival, le FPA-GA a donné de bons résultats par rapport à son algorithme rival et finalement une comparaison entre les deux algorithmes concurrents n'est pas réalisable en utilisant l'outil de test de Wilcoxon.

Ainsi, la moyenne et l'écart-type ($Mean \pm Standard\ deviation$) obtenus, pour chaque algorithme et pour chaque fonction, sur les 25exécutions seront illustrées. Il est à noter aussi que, la dimension de tous les problèmes traités (toutes les fonctions benchmarks) est fixée à 30.

5.5Résultats et discussions :

Les valeurs moyennes des optimums globaux et l'écart type (*std*.) générés par les algorithmes courants, sont illustrés dans le tableau 5.1. Pour plus de clarté, les cases des résultats obtenues, qui sont de meilleure qualité, sont mises en couleur grise.

Par conséquent, nous pouvons observer clairement à partir de ce tableau que FPA-GA a obtenu de meilleurs résultats par rapport aux autres algorithmes. Plus précisément (voir le tableau 5.2), les FPA-GA a obtenu de meilleurs résultats que les FPA, MGOFPA, CMA-ES, CLPSO, JADE, jDE et CoDE dans respectivement 24, 24, 21, 23, 20, 23 et 21 cas (fonctions)de 25 cas traité et égal à ces derniers dans 1, 1,1,1,2,1,2 sur les 25 traité. En outre, FPA-GA fonctionne moins bien dans 3, 1, 3, 1,2 sur 25 cas que CMA- ES, CLPSO, JADE, jDE et CoDE respectivement.

Fonction	FPA-GA		FPA		MGOFPA		CMA-ES		CLPSO		JADE		jDE		CoDE	
	Mean	**Std Dev**	**Mean**	**Std Dev**	**Mean**	**Std Dev**	**Mean**	**Std Dev**	**Mean**	**Std Dev**	**Mean**	**Std Dev**	**Mean**	**Std Dev**	**Mean**	**Std Dev**
1	0.00e+00	0.00e+00	4.03e-29	1.08e-28	1.26e-29	4.59e-29	1.80e-25	4.65e-26	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
2	3.38e-11	7.54e-11	3.44e-18	8.02e-18	1.55e-02	1.79e-02	6.37e-25	1.78e-25	8.12e+02	2.32e+02	9.37e-29	1.07e-28	8.84e-07	1.12e-06	1.77e-15	2.19e-15
3	3.77e-10	1.65e-10	1.38e+06	1.87e+06	1.27e+06	8.95e+05	5.13e-21	1.30e-21	1.70e+07	2.61e+06	6.57e+03	3.64e+03	2.02e+05	9.92e+04	1.08e+05	5.38e+04
4	4.66e-06	1.15e-06	2.46e-04	4.79e-04	3.73e-01	3.87e-01	6.11e+05	1.68e+06	6.79e+03	1.10e+03	2.56e-14	8.55e-14	3.07e-02	5.68e-02	8.09e-03	2.24e-02
5	1.64e-04	1.27e-04	5.92e+01	2.21e+01	4.80e+01	7.00e+00	3.35e-10	8.62e-11	4.13e+03	4.76e+02	7.89e-06	3.52e-05	5.65e+02	5.22e+02	5.14e+02	4.42e+02
6	1.07e-11	4.12e-12	2.16e+01	4.26e+01	1.75e+01	2.00e+01	3.98e-01	1.22e+00	5.90e+00	1.27e+01	1.00e+01	2.85e+01	2.47e+01	2.69e+01	7.39e-10	1.99e-09
7	8.32e-05	5.38e-06	1.85e-02	1.43e-02	2.68e-02	3.52e-02	1.81e-03	4.39e-03	4.48e-01	8.44e-02	8.17e-03	7.32e-03	1.19e-02	7.76e-03	7.41e-03	8.51e-03
8	4.67e-02	1.46e-04	2.10e+01	9.79e-02	2.10e+01	7.65e-02	2.04e+01	6.89e-01	2.09e+01	5.05e-02	2.09e+01	6.48e-02	2.09e+01	3.31e-02	2.01e+01	1.05e-01
9	1.76e-03	1.42e-04	5.27e+01	2.38e+01	2.55e+01	9.76e+00	4.00e+02	1.15e+02	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
10	1.13e-03	1.47e-04	1.15e+02	8.63e+01	4.36e+01	3.58e+01	4.41e+01	1.49e+01	1.04e+02	1.77e+01	2.25e+01	2.96e+00	5.33e+01	8.70e+00	3.82e+01	1.14e+01
11	2.00e-02	1.04e-03	3.29e+01	8.41e+00	2.12e+01	7.49e+00	6.72e+00	2.23e+00	2.60e+01	1.72e+00	2.54e+01	2.27e+00	2.76e+01	1.46e+00	1.32e+01	3.84e+00
12	2.55e-02	1.54e-03	4.19e+01	7.61e+00	3.19e+01	8.43e+00	1.36e+04	1.42e+04	1.95e+04	5.56e+03	6.30e+03	7.21e+03	6.05e+03	5.30e+03	2.63e+03	1.91e+03
13	5.83e-04	2.58e-04	7.50e+00	7.29e+00	3.91e+00	2.97e+00	3.22e+00	8.63e-01	2.10e+00	2.21e-01	1.51e+00	6.68e-02	1.68e+00	1.21e-01	1.56e+00	3.17e-01
14	6.88e-02	7.40e-04	1.34e+01	5.76e-01	1.35e+01	2.98e-01	1.47e+01	2.33e-01	1.27e+01	2.28e-01	1.22e+01	2.76e-01	1.29e+01	2.20e-01	1.24e+01	4.91e-01
15	7.51e-04	4.35e-05	2.53e+02	8.60e+01	3.00e+02	1.12e+02	3.67e+02	2.10e+02	6.12e+01	4.10e+01	3.51e+02	1.14e+02	3.89e+02	8.78e+01	4.00e+02	7.94e+01
16	7.87e-04	1.05e-04	2.66e+02	1.15e+02	9.27e+01	8.02e+01	3.65e+02	3.10e+02	1.70e+02	3.47e+01	1.28e+02	1.42e+02	7.81e+01	2.23e+01	6.45e+01	1.64e+01
17	7.58e-04	1.29e-04	1.90e+02	1.29e+02	2.68e+02	8.71e+01	4.97e+02	3.47e+02	2.54e+02	4.25e+01	1.18e+02	1.00e+02	1.46e+02	4.27e+01	6.51e+01	1.27e+01
18	6.82e-04	2.44e-05	8.26e+02	2.01e+00	9.04e+02	1.07e+00	9.03e+02	2.12e-01	9.14e+02	1.34e+00	9.03e+02	2.61e-01	9.04e+02	1.23e+00	9.04e+02	9.64e-01
19	6.86e-04	2.96e-05	8.25e+02	1.77e+00	2.12e+02	3.13e+00	9.03e+02	2.12e-01	9.09e+02	2.20e+01	9.04e+02	1.06e+00	9.04e+02	1.04e+00	9.04e+02	1.04e+00
20	6.96e-04	3.14e-05	8.25e+02	2.28e+00	9.03e+02	7.08e-01	9.03e+02	2.99e-01	9.12e+02	8.11e+00	9.04e+02	7.62e-01	9.04e+02	1.07e+00	9.04e+02	1.34e+00
21	6.23e-04	1.69e-05	7.39e+02	1.80e+02	5.33e+02	1.32e+02	5.00e+02	2.63e-12	5.00e+02	1.25e-12	5.00e+02	5.05e-14	5.00e+02	3.91e-14	5.00e+02	8.65e-14
22	4.71e-04	4.94e-05	5.08e+02	4.57e+00	8.74e+02	2.22e+01	8.19e+02	1.30e+01	9.64e+02	1.05e+01	8.66e+02	2.05e+01	8.74e+02	1.54e+01	8.58e+02	2.23e+01
23	6.31e-04	2.32e-05	7.88e+02	1.83e+02	5.90e+02	1.73e+02	5.36e+02	3.89e+00	5.34e+02	2.04e-04	5.54e+02	9.00e+01	5.34e+02	2.59e-04	5.34e+02	4.51e-04
24	6.17e-04	1.53e-05	2.12e+02	3.13e+00	5.77e+02	3.57e+02	2.00e+02	6.18e-14	2.00e+02	1.46e-12	2.00e+02	2.91e-14	2.00e+02	2.91e-14	2.00e+02	2.91e-14
25	5.05e-04	1.05e-05	2.16e+02	6.91e-01	1.56e+03	1.09e+01	2.10e+02	6.05e+00	2.00E+02	1.96E+00	2.13e+02	7.95e-01	2.11e+02	7.31e-01	2.13e+02	9.12e-01

Table 5.1. Résultats de simulation obtenue relatifs aux algorithmes FPA, MGOFPA, CMA-ES, CLPSO, JADE, jDE, CoDE et FPA-GA pour 25 essais indépendants (chaque essai a un nombre d'itérations= Max_Obj_Ev) et une dimension de problèmes traités égale 30.

Fonction	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	⊖	⊙	⊕
Algorithms																												
FPA	⊖	⊙	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	**24**	**1**	**0**
MGOFPA	⊙	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	**24**	**1**	**0**
CMA-ES	⊙	⊕	⊕	⊖	⊕	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	**21**	**1**	**3**
CLPSO	⊙	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊕	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	**23**	**1**	**1**
JADE	⊙	⊙	⊖	⊕	⊕	⊖	⊖	⊖	⊕	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	**20**	**2**	**3**
JDE	⊙	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊕	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	**23**	**1**	**1**
CoDE	⊙	⊕	⊖	⊖	⊖	⊙	⊖	⊖	⊕	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	⊖	**21**	**2**	**2**

Table 5.2. Résumé de comparaison en utilisant l'outil de test Wilcoxon rank-sum entre l'algorithme proposé FPA-GA (considéré ici comme une référence) et les autres algorithmes concurrents.

Un petit coup d'œil sur les résultats obtenus (voir tableaux 5.1 et 5.2), montre que l'intégration des deux opérateurs (croisement et mutation) comme des éléments principaux dans la structure algorithmique de FPA, permet d'améliorer considérablement sa performance. En effet, ceci est dû principalement à une amélioration significative de la caractéristique de *diversification*, puisque à chaque itération t (surtout lorsque la condition (4.1) est vérifiée) trois sous-population seront générées, chose qui augmente significativement les chances de trouver une meilleure solution. Il est à noter aussi que, la caractéristique d'*intensification* d'algorithme FPA est aussi amélioré par effet d'association de ces deux opérateurs ainsi que le mécanisme de la sélection élite déjà existé.

5.6 Conclusion :

Dans ce chapitre, nous avons présenté en détail l'outil de test utilisé pour prouver l'efficacité de l'algorithme proposé. En réalité, cet outil est présenté par un ensemble de fonctions qui sont définies dans le cadre de la conférence 2005 *IEEE Congress on Evolutionary Computation* (CEC 2005) [56]. L'examen de l'algorithme proposé est réalisé à travers des essais de simulation, en comparant ses résultats avec ceux obtenus en employant d'autres méthodes d'optimisation. Plus précisément, une étude comparative a été réalisée en basant sur ; (i) une observation directe des résultats obtenus (voir tableau 5.1) et sur (ii) les résultats tirés d'une analyse statistique en utilisant le test dit ; *Wilcoxon rank-sum*.

Comme nous l'avons dit, les modifications introduites sur le FPA (sous sa structure de base) par l'intégration des deux opérateurs de GA à savoir ; l'opérateur de croisement et ce de mutation, ont permis d'améliorer significativement ses performances en termes d'augmentations au sens positif de ses deux caractéristiques ; *intensification* et *diversification*.

Conclusion générale

Conclusion générale

Les travaux de recherche présentés dans ce mémoire de master concernent le développement et l'application d'une nouvelle méthode d'optimisation de type combinatoires mono-objectifs s'appuyant sur les algorithmes métaheuristiques. Nous avons focalisé, nos travaux sur ce type d'algorithmes destinés à résoudre des problèmes d'optimisation difficiles.

Nous avons présenté, dans un premier temps, une petite introduction descriptive sur les métaheuristiques à population de solutions dédiées aux problèmes d'optimisation combinatoire mono-objectifs au sens général.

Un intérêt particulier a été donné à une méthode d'optimisation dite ; algorithme de pollinisation des fleurs (FPA). En effet, la version primitive de cet algorithme, présente comme la majorité de toutes les méthodes appartenant à sa classe sous leurs versions standards, des insuffisances majeures telles que leurs intensifications et diversification (relativement) faibles.

Un travail d'amélioration de ce cette méthode a été proposé.Plus précisément, nous avons présenté une nouvelle variante relative à FPA, afin d'améliorer principalement ses deux caractéristiques ; intensification et diversification. Les objectifs visés, ont été réalisés en intégrant trois opérateurs appartenant aux AGs dans la corp algorithmique de FPA, ces derniers sont respectivement ; l'opérateur de croisement et celui de mutation suivis tous les deux par une opération de sélection dite (en anglais) ; Elite Sélection.

L'algorithme proposé, qui porte le nom FPA-GA a montré de bonnes performances en termes de qualité des solutions trouvées. En effet, il permet d'obtenir des résultats qui dépassent de manière très significative ceux obtenus par la version standard du FPA et d'autres algorithmes appartenant à la même famille de ce dernier.

L'algorithme proposé a été appliqué sur un ensemble de fonctions de tests dites ; fonctions benchmarks définies dans le cadre de la conférence 2005-IEEE Congress on Evolutionary Computation (CEC 2005).

À partir des analyses et des expérimentations faites sur ces fonctions, nous avons conclu que l'amélioration introduite par l'hybridation du FPA et un algorithme GA simple, a permis de renforcer -spécialement- au sens positif les deux caractéristiques intensification et diversification.

En perspective, une analyse expérimentale plus poussée reste à faire. Les paramètres d'algorithme proposé peuvent également être auto-adaptés, afin de faciliter leur utilisation. De plus, l'algorithme proposé dans ce mémoire a le type mono-objectif, il ne peut optimiser qu'un

Seul fonction objective à la fois. Néanmoins, de nombreux problèmes de calcul des paramètres dans certains problèmes nécessitent l'optimisation de plusieurs fonctions objectifs simultanément.

Références bibliographiques

Références bibliographiques

[1] X.S. Yang. "Flower pollination algorithm for global optimization, "In: Unconventional Computation and Natural Computation, Springer, Berlin, 2012, pp.240–249.

[2] J. H. Holland. "Adaptation in Natural and Artificial Systems ". The University of Michigan Press, 1975.

[3] R. G. Reynolds. "An introduction to cultural algorithms ". In A. V. Sebald& L. J. Fogel (Eds.), Proceedings of the Third Annual Conference on Evolutionary Programming, Singapore, pp. 131- 139. 1994.

[4] C. H. Papadimitriou and K. Steiglitz. "Combinatorial optimization: algorithms and complexity ". Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1982.

[5] S. Kirkpatrick, C. Gelatt, and M. Vecchi. "Optimization by simulated annealing». Science, Vol. 220, No. 4598, pp. 671–680, 1983.

[6] F. Glover. "Future paths for integer programming and links to artificial intelligence ". Computers and Operations Research, Vol. 13, No. 5, pp. 533–549, 1986.

[7] T. A. Feo and M. G. C. Resende. "A probabilistic heuristic for a computation allydifficult Set covering problem ". Operations Research Letters, Vol. 8, No. 2, pp. 67– 71, 1989.

[8] T. A. Feo and M. G. C. Resende. "Greed yrandomized adaptive search procedures ". Journal of Global Optimization, Vol. 6 No. 2, pp. 109–133, 1995.

[9] N. Mladenovic. "A variable neighborhood algorithm - a new metaheuristic for combinatorial optimization ". In Abstracts of papers presented at Optimization Days, pp112, Montréal, Canada, May 1995.

[10] N. Mladenovic and P. Hansen. "Variable neighborhood search ". Computers and Operations Research, Vol. 24, pp. 1097–1100, 1997.

[11] T. Stützle. "Local Search Algorithms for Combinatorial Problems: Analysis, Improvements, and New Applications ". PhD thesis, Darmstadt University of Technology, 1998.

[12] I. Boussaïd. "Perfectionnement de métaheuristiques pour l'optimisation continue". Thèse de Doctorat, université des sciences et de la technologie Houari Boumediene (USTHB), Algérie, 2013.

[13] I. Boussaïd, J. Lepagnot, and P. Siarry. "A survey on optimization metaheuristics Information Sciences, Vol. 237, pp. 8–117, 2013.

[14] M. Dorigo and L. M. Gambardella. "Ant Colonies for the Traveling Salesman Problem ". BioSystems, Vol. 43, pp. 73–81. 1997.

[15] Z. Yunqiang, T. Ying, S. Chaoli, and Z. Jianchao. ''a Hybrid Intelligent Algorithm for Mixed Variable Optimization Problems ''. Future Communication, Computing, Control and Management Lecture Notes in Electrical Engineering, Vol. 141, pp 249-256, 2012.

[16] W.E. Combs. "Combinatorial rule explosion eliminated by a fuzzy rule configuration ". IEEE Transactions on Fuzzy Systems, Vol. 6, No. 1, pp. 1 – 11, 1998.

[17] C. Darwin. "On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life ". J. Murray, June 1859.

[18] T. Bäck, U. Hammel, and F. P. Schweffel. "Evolutionary Computation: Comments on the history and current state ". IEEE transactions on Evolutionary Computation, Vol. 1, No. 1, pp.33- 17,1997.

[19] J. H. Holland. “Adaptation in Natural and Artificial Systems “. The University of Michigan Press, 1975.
[20] J. R. Koza. “Hierarchical genetic algorithms operating on populations of computer programs”. pp. 768–774, Morgan Kaufmann, Detroit, Michigan, USA, 1989.
[21] J. R. Koza. “Genetic programming: A paradigm for genetically breeding populations of computer programs to solve problems”. Technical Report, STANCS-90-1314, Stanford University, Department of Computer Science, 1990.
[22] I. Rechenberg. “Cybernetic solution path of an experimental problem”. Technical report, Library translation 1122, Ministry of Aviation, Royal Air Force Establishment (UK), 1965.
[23] L. J. Fogel. “Autonomous automata”. Industrial Research Magazine, Vol. 4, No. 2, pp. 14–19, 1962.
[24] L. J. Fogel, A. J. Owens, and M. J.Walsh. “Artificial Intelligence through Simulated Evolution“ John Wiley& sons, first edition, 1966.
[25] E. Bonabeau. M. Dorigo, and G. Theraulaz. “Swarm intelligence: from natural to artificial systems “. Oxford University Press, Inc., New York, NY, USA, 1999.
[26] J. Kennedy. And R. C. Eberhart. “Particle SwarmOptimization”. In: Proceedings of the IEEE International Conference on Neural Networks IV, pp. 1942–1948, Perth, Australia, November, 1995.
[27] K. M. Passino. “Biomimicry of bacterial foraging for distributed optimization and control “. IEEE Control Systems Magazine, Vol. 22, No. 3, pp. 52–67, 2002.
[28] D. Karaboga, and B. Basturk. “A powerful and efficient algorithm for numerical function optimization: artificial bee colony algorithm“.Journal of Global Optimization. Vol. 3, pp. 9459– 9471, 2007.
[29] M. Dorigo, V. Maniezzo, and A. Colorni. “Positive feedback as a search strategy “. Technical Report 91-016, Dipartimentodid Elettronica, Politecnicodid Milano, Milan, Italy, 1991.
[30] M. Dorigo. “Optimization, Learning and Natural Algorithms “. PhD thesis, Politecnico di Milano, Italy, 1992.
[31] M. Dorigo, V. Maniezzo, and A. Colorni. “Ant system: optimization by a colony of cooperating agents “. IEEE Transactions on Systems, Man, and Cybernetics, Part B, Vol. 26 No. 1, pp. 29–41, 1996.
[32] Dubey, Hari Mohan, Manjaree Pandit, and B. K. Panigrahi. "Hybrid flower pollination Algorithm with time-varying fuzzyselectionmechanism for wind integrated multi-Objective dynamic economic dispatch." Rene wable Energy 83 (2015): 188-60 202.
[33] Abdel-Baset M, Hezam IM (2015) an improved flower pollination algorithm for ratios Optimization problems. App. Math INF Sci Lett3 (2):83–91.
[34] Rodriguez D, Yang XS, De Souza AN, Papa JP (2015) Binary flower pollination Algorithm and its application to feature selection. In: Recent advances in swarm Intelligence and evolutionary computation. Springer, pp 85–100.
[35] Bensouyad M, Saidouni DE (2015) A hybrid discrete flower pollination algorithm for Graph coloring problem. In Proceedings of the international conference on Engineering & MIS 2015. ACM, p 22.
[36] Mishra A, Deb S (2016) AssemblysequenceoptimizationusingflowerpollinationAlgorithm-basedapproach. J IntellManuf. https://doi.org/10.1007/s10845-016-1261-7
[37] Yang XS, Karamanoglu M, He X (2013a) Multi-objective flower algorithm for Optimization. Proc ComputSci18:861–868.

[38] Yang XS, Karamanoglu M, He X (2014) Flower pollination algorithm: anovelapproach For multi objective optimization. Eng Optim 46(9):1222–1237.
[39] Nabil E (2016) A modified flower pollination algorithm for global optimization. Expert Syst A pp l57:192–203.
[40] Kalra S, Arora S (2016) Firefly algorithm hybridized with flower pollination algorithm for multimodal functions. In: Proceedings of the international congress on information and communication technology. Springer, Singapore, pp 207–219.
[41] Yang X-S, Suash D (2010) Eaglestrategyusing Lévy walk and firefly algorithms for stochastic optimization. In: Nature inspired cooperative strategies for optimization (NICSO 2010). Springer, Berlin, pp 101–111.
[42] Abdel-Basset, M., Shaky, L.A.: Flower pollination algorithm: a comprehensive review. Artif. Intel. Rev. 1–25 (2018).
[43] E. Talbi. "Metaheuristics: from design to implementation ". John Wiley and Sons, NJ, USA, 2009.
[44] C. R. Reeves, and J. E. Rowe. "Genetic algorithms: Principles and Perspectives ". Kluwers Academic Publishers, NY, USA, 2002.
[45] M. Gen and R. Cheng. "Genetic Algorithms and engineering optimization ". Wiley inter-Science, 2000.
[46] T. Bäck, D.F. Fogel, and Z. Michalewicz. "Evolutionary Computation 1 Basic Algorithms and Operators ".
[47] F. G. Lobo and C. F. Lima. "A review of adaptive population sizing schemes in genetic algorithms ". In Proceedings of the 2005 Workshops on Genetic and Evolutionary Computation, pp. 228–234, 2005.
[48] T. Bäck, D.F. Fogel, and Z. Michalewicz. "Handbook of Evolutionary Computation ". IOP Publishing Ltd and Oxford University Press, PA, USA, 1997.
[49] K. Jebari, A. Bouroumi, and A. Ettouhami. "An experimental study of parent selection Operators for genetic algorithms ". In Proceedings of International Conference on Software, Knowledge Information Management and Applications, pp. 56–61, Fez, Morocco, 2009.
[50] F. Herrera, M. Lozano, and A.M. Sanchez. "A taxonomy for the crossover perator for Real-coded genetic algorithms: An experimental study". International Journal of Intelligent Systems, Vol. 18, pp. 309–338, 2003.
[51] T. Bäck, D.F. Fogel, and Z. Michalewicz. "Evolutionary Computation 1 Basic Algorithms and Operators ". Institute of Physics Publishing, PA, USA, 2000.
[52] N.M. Worthy, and W. M. Spears. "Foundations of Genetic Algorithms ". Academic Press, London, UK, 2001.
[53] A.E. Eiben, and J.E. Smith, "Introduction to Evolutionary Computing", Assembly Automation, Automation, Vol. 24, No. 3, pp. 324–324, 2004.
[54] A.P. Engelbrecht." Computational Intelligence; An introduction", Second Edition, John Wiely, University of Pretoria, South Africa, 2007.
[55] R. Kumar, G.Gopal &R.Kumar, "Hybridization in Genetic Algorithms, "vol-3, issue-2, IJARCSSE2011, pp252-258.
[56] A. Auger, and S. Tiwari, "Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization," Nanyang Technol. Univ., Singapore, Tech. Rep. KanGAL #2005005, May 2005, IIT Kanpur, India.

[57] Draa A (2015) on the performances of the flower pollination algorithm–Qualitative and quantitative analyses. Appl Soft Comput 34:349–371.

[58] N. Hansen and A. Ostermeier. “Complete lyderandomized self-adaptation in evolution strategies,” Evolut. Comput. vol. 9, no. 2, pp. 159–195, 2001.

[59] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar. “Comprehensive learning particle swarm optimizer for global optimization of multimodal functions,” IEEE Trans. Evolut. Comput., vol. 10, no. 3, pp. 281–295, 2006.

[60] J. Zhang and A. C. Sanderson. “JADE: adaptive differential evolutionwithoptionalexternal archive,” IEEE Trans. Evolut. Comput, vol. 13, no. 5, pp. 945-958, 2009.

[61] 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. Evolut. Comput, vol. 10, no. 6, pp. 646–657, Dec. 2006.

[62] 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, Feb. 2011.