Métaheuristiques adaptatives d'optimisation continue basées sur des méthodes d'apprentissage

Asmaa Ghoumari

UNIVERSITE PARIS EST

THESE DE DOCTORAT
EN INFORMATIQUE

École doctorale de Mathématiques et STIC (MSTIC, ED 532)
Laboratoire Images, Signaux et Systèmes Intelligents (LISSI, EA-3956)

Métaheuristiques adaptatives d'optimisation continue basées sur des méthodes d'apprentissage

---

par

Asmaa GHOUMARI

Thèse dirigée par **Amir NAKIB et Patrick SIARRY**

Soutenue le XX décembre 2018

Jury :

Rapporteur	Jin-Kao	HAO	Maître de Conférences	Université d'Angers
Rapporteur	Farouk	YALAOUI	Professeur des Universités	Université de Technologie de Troyes
Examinatrice	Anne	AUGER	Chargée de Recherche	INRIA
Examinateur	Amir	NAKIB	Maître de Conférences	Université Paris-Est Créteil
Examinateur	Patrick	SIARRY	Professeur des Universités	Université Paris-Est Créteil

Introduction générale

Les scientifiques sont confrontés à des problèmes extrèmement variés, dont la complexité ne cesse d'augmenter avec les années. En optimisation, les problèmes sont classés en plusieurs grandes catégories : discret ou continu, statique ou dynamique, mono-objectif ou multi-objectif, etc. Ces catégories permettent d'élaborer des méthodes de résolution qui vont être adaptées aux propriétés intrinsèques des problèmes afin d'être les plus performantes possibles. En optimisation, un problème est appelé fonction objectif, ou fonction de coût, et le but va être de minimiser ou de maximiser la fonction considérée, c'est-à-dire de trouver l'optimum global quand il existe (minimum ou maximum), avec ou sans contraintes à respecter. Pour illustrer ce qu'est un problème d'optimisation, on peut prendre pour exemple les entreprises qui veulent minimiser les coûts de production, tout en maximisant leurs profits, ou encore les circuits électroniques qui doivent maximiser leur rendement et les performances. Il existe de nombreux de problèmes d'optimisation appartenant à des domaines très variés allant du traitement d'image, de l'informatique, de l'industrie, de la physique, etc.

Les méthodes de résolution vont parcourir l'espace des solutions, ou espace de recherche, afin de trouver la meilleure solution possible. Elles requièrent souvent des processus itératifs qui vont améliorer une ou plusieurs solutions à la fois. Ainsi, l'espace de recherche va être parcouru et la recherche s'orientera au fur et à mesure vers la solution recherchée. Afin de résoudre ces problèmes d'optimisation, des méthodes ont été élaborées au fil des années, pour notamment aboutir aujourd'hui à deux grandes familles de méthodes, qui sont d'ailleurs apparues respectivement l'une après l'autre : les heuristiques et les métaheuristiques. Les heuristiques sont des méthodes conçues pour résoudre spécifiquement un type de problème, alors que les métaheuristiques visent à résoudre non pas un problème spécifiquement mais une gamme plus large de problèmes. Dorénavant, les métaheuristiques prospèrent davantage, ce que l'on peut mettre en corrélation avec l'augmentation de la puissance de calcul des ordinateurs ainsi qu'à leur meilleurs résultats sur des problèmes d'optimisation toujours plus complexes.

Parmis les métaheuristiques les plus populaires, on retrouve les algorithmes basés sur des populations de solutions (type algorithmes d'essaims ou algorithmes évolutionnaires), en opposition avec les algorithmes à une seule solution (tels que le recuit simulé, la descente de gradient, la recherche tabou, etc).

Dans cette thèse, on s'intéresse plus particulièrement aux algorithmes évolutionnaires (EA). Ils simulent le processus d'évolution naturelle définie par Charles Darwin au 19ème siècle, d'où leur qualification de *bioinspirés*. Un algorithme évolutionnaire est constitué d'un ensemble de solutions, appelé *population d'individus*, qu'il va s'agir de faire évoluer au fur et à mesure des itérations, ici des *générations*, grâce à des opérateurs d'évolution afin qu'au moins un individu de la population évalué par la fonction objectif atteigne l'optimum global de celle-ci.
Il va donc s'agir d'avoir une vue globales des solutions à fort potentiel dans l'espace de recherche, une sorte de point de vue global, et puis ensuite de approfondir la recherche dans des zones locales pour déboucher vers la meilleure solution possible. Ces deux échelles à la fois globale et locale sont définies par deux phases : l'exploration (ou diversification) et l'exploitation (ou intensification).
Plus spécifiquement, l'enjeu pour les algorithmes évolutionnaires est de maintenir l'équilibre entre l'intensification et l'exploration car il est très compliqué pour eux une fois l'intensification avancée de pouvoir par la suite repartir explorer d'autres régions. Une métrique est alors utilisée pour mesurer la répartition des individus dans l'espace de recherche : la diversité. Concernant les EA, on observe une baisse de la diversité précipée dès les premières générations, ce qui les rend sensibles aux optima locaux. Pour pallier cette faiblesse, il faut donc être attentif aux opérateurs d'évolution employés car ils impactent sur la diversité de la population, et donc sur les performances des EA. Or, dans la littérature il existe de nombreux opérateurs, et il devient alors compliqué pour l'utilisateur de concevoir un EA adapté qui résoudra son problème d'optimisation.

Il devient alors intéressant de contribuer à concevoir des EA plus simples à concevoir pour l'utilisateur, et permettant d'appliquer les opérateurs adaptés à chaque instant de la recherche pour améliorer les performances. On s'est donc penchés sur les algorithmes adaptatifs, capables de modifier leur comportement de manière dyamique. Dans cette thèse, nous proposons deux algorithmes évolutionnaires adapta-

tifs, d'abord le *maximum a posteriori based evolutionary algorithm* (MEA), puis le *Evolutionary Algorithm based on a Dynamic Graph* (EADG). Les deux approches apportent une nouvelle couche de gestion des opérateurs, reposant chacune sur des méthodes différentes.
Afin d'appliquer les opérateurs permettant de préserver au maximum la diversité, il existe des paramètres communs à ces deux algorithmes : le nombre de stratégies N, et la longueur des intervalles de Δ générations, appelés moments de décision. Une stratégie est un couple composé d'un opérateur de croisement avec un opérateur de mutation. La diversité est mesurée grâce à la distance euclidienne entre les individus de la population.
Le MEA s'articule autour du principe du maximum a posteriori, en mesurant les diversités passées de chaque stratégie, on prédit les probabilités de diversités à venir, et ainsi on peut choisir la stratégie maximisant la diversité future. On parle alors de mécanisme de prédiction.
Le EADG se base sur un graphe de stratégies. Les noeuds correspondent aux stratégies, et les arcs reliant les noeuds possèdent des poids calculés sous forme de probabilités à partir des diversités passées. Au cours de la recherche de l'optimum, l'algorithme va parcourir le graphe en passant d'un noeud à un autre, et donc en appliquant une stratégie puis une autre. La sélection d'une stratégie se fait en considérant les poids sur les arcs qui sont mis à jour tous les Δ générations après la mesure des diversités passées. Là aussi il s'agit d'un mécanisme d'apprentissage. De plus, un modèle de substitution via un réseau de neurones, est mis en place afin de pouvoir simuler la fonction objectif en prenant en compte tous les individus rencontrés tout au long de la recherche.

Cette thèse a été préparée au Laboratoire Images, Signaux et Systèmes Intelligents (LISSI, EA-3956), et appartenant à l'école doctorale Mathématiques et STIC (MSTIC, ED 532) de l'Université Paris-Est (UPE). Elle a été dirigée par Dr Amir Nakib, maître de conférences, et Pr Patrick Siarry, professeur des Universités, au sein groupe SIMO (SIgnal, IMage et Optimisation).
Ce travail va se concentrer sur les méthodes adaptatives, en particulier celles appartenant à la famille des algorithmes évolutionnaires, et les associées à des méthodes d'apprentissage. Cette thèse aboutit à la proposition de deux nouvelles métaheuristiques mêlant les concepts précédemment évoqués, et l'analyse de leurs performances

sur un benchmark d'optimisation continue.
Les contributions de ce travail de thèse sont :

- la conception de nouvel algorithme performant, MEA, adapté aux problèmes d'optimisation continue ;
- la conception de nouvel algorithme performant, EADG, adapté aux problèmes d'optimisation continue ;

Le manuscrit s'articule autour de quatre chapitres, le plan est détaillé ci-dessous.
Dans le premier chapitre, nous présentons un état de l'art les métaheuristques d'optimisation continue, dont les méthodes adaptatives d'optimisation.
Le deuxième chapitre présente le premier algorithme proposé, le *Maximum a posteriori Evolutionary Algorithm* (MEA). Cette nouvelle métaheuristique permet d'articuler un algorithme évolutionnaire avec le principe du maximum a posteriori *MAP* afin de le rendre adaptatif. Nous commençons le chapitre par décrire l'architecture et le fonctionnement du MEA. Puis, nous étudions les sensibilités des paramètres, c'est-à-dire l'impact du paramétrage sur les performances de l'algorithme.
Dans le troisième chapitre, nous décrivons le second algorithme mis au point, EADG. Cette méthode propose aussi un algorithme évolutionnaire adaptatif mais en se reposant sur un graphe dynamique d'opérateurs. Ce dernier s'utilise aussi sur un réseau de neurones LSTM permettant d'approximer la fonction objectif en créant un modèle de celle-ci. Nous commençons par définir les réseaux de neurones LSTM, puis nous décrivons la structure ainsi que le déroulement de l'algorithme. Enfin, nous effectuons une étude des sensibilités des paramètres de EADG.
Dans le quatrième chapitre, nous analysons les performances de MEA et EADG, les comparant à ceux obtenus par 13 autres algorithmes sur un ensemble de 34 fonctions d'optimisation continue, puis nous observons les résultats obtenus pour le problème du cluster atomique de Lennard-Jones, et enfin nous étudions leur complexité respective en les comparant à celle d'un algorithme évolutionnaire classique.
Pour clore ce manuscrit, une conclusion générale résume les points principaux qui auront été présentés, et les travaux futurs à envisager comme perspectives.

Chapitre 1

Etat de l'art en optimisation continue

1.1 Introduction

L'optimisation est issue des mathématiques et connait un essor ces dernièrs siècles qui est notamment dû à la multitude d'applications possibles depuis le 20 ème siècle avec l'industrialisation des pays développés, puis aux nouvelles sciences du traitement de signal, de planification, d'automatique, d'aide à la décision, etc, et plus récemment de l'intelligence artificielle.
Concernant la formulation des problèmes d'optimisation, les premiers remontent à Euclide au 3 ème siècle avant Jésus-Christ et sont formulés dans son livre *Les Eléments*. Puis 300 ans plus tard, Héron d'Alexandrie définit en optique le principe du plus court chemin. En effet, l'optimisation a continué d'évoluer suite aux travaux de De Fermat (mort en 1665), Lagrange (1736-1813) et Sir Hamilton (1805-1865), puis sont arrivées les méthodes itératives élaborées par Newton (1643-1727) et Gauss (1777-1855) et Leibniz(1646-1716), la mécanique newtonienne qui va d'ailleurs mettre en avant de nouvelles méthodes d'optimisation. Toujours au 17 ème siècle, Euler (1707-1783) et Lagrange développent en analyse fonctionnelle le calcul variationnel, qui regroupe un ensemble de méthodes permettant de minimiser une fonction dans un espace vectoriel. Ensuite, les méthodes de programmation linéaire sont arrivées grâce au travail de Kantorovich (1912-1986) [62] puis elles ont été améliorées par Dantzig (1914-2005) [25] (méthode du simplexe). C'est d'ailleurs ce dernier qui a donné en 1947 le terme de "programmation linéaire" en faisant référence à la planification de programmes militaires pendant la seconde guerre mondiale et non à la programmation informatique. Aujourd'hui le terme de méthodes d'optimisation linéaire est davantage utilisé.
Depuis les années 50, l'optimisation s'est largement développée, et il y a aujourd'hui

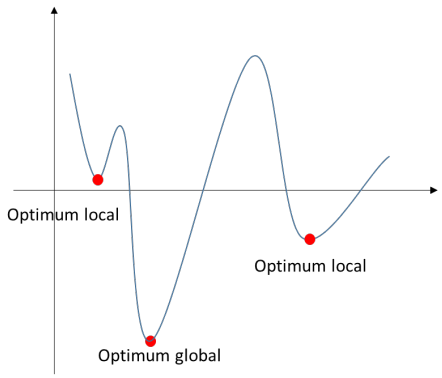


Figure 1.1 – Illustration dans le cas d'un problème d'optimisation à minimiser, mise en exergue d'optima locaux et global.

plusieurs grandes familles de méthodes séparables en différentes catégories : discret ou continu, mono-objectif ou mulit-objectif, stochastique ou déterministe, etc. Parmis les types de méthodes les plus connues et utilisées à ce jour : les heuristiques et les métaheuristiques. Ces familles d'algorithmes seront présentées et étudiées dans ce chapitre.

Un problème d'optimisation consiste à minimiser, ou maximiser, une fonction mathématique f, appelée fonction objectif ou fonction de coût. Le but est de trouver la (ou les) solution(s) optimale(s) x^* parmi un ensemble de solutions, noté S, appelé espace de recherche ou espace de solutions, et tel que $S \in \mathbb{R}$ l'ensemble des réels. Mathématiquement un problème de minimisation est noté ainsi :

$$f(x^*) \leq f(x), \forall x \in S, soit \min_{x \in S} f(x) \tag{1.1}$$

De même, pour un problème de maximisation :

$$f(x^*) \geq f(x), \forall x \in S, soit \max_{x \in S} f(x) \tag{1.2}$$

L'enjeu consiste alors à trouver l'optimum global, qui est la meilleure solution possible x^* de la fonction de coût f. Afin d'illustrer cela, la figure Fig.1.1 montre des optima locaux et un optimum global d'une fontion de coût.

Pour rappel, un optimum local est défini dans un cas de minimisation tel qu'un point α admettant un voisinage $V \in S$ tel que $f(\alpha) \leq f(x)$, $\forall x \in V$, (inversement dans le cas d'une maximisation). Cependant il peut être remarqué que les fonctions mathématiques ont rarement un seul optimum, mais plutôt plusieurs optima locaux dont un optimum global, qui peut être parfois difficile à atteindre pour un algorithme et pouvant ainsi se retrouver coincé dans un optimum local suivant la rugosité du paysage (modèle du problème dans l'espace de recherche).

Ce premier chapitre présente un état de l'art de l'optimisation continue en lien avec le travail présenté dans cette thèse constituée d'algorithmes évolutionnaires adaptatifs. Dans la première partie, nous définissons ce qu'est un problème d'optimisation, puis nous présentons les heuristiques et les métaheuristiques, enfin nous détaillons les métaheuristiques à solution unique, puis à population de solutions dont les algorithmes évolutionnaires.

1.2 Heuristiques et métaheuristiques

Pour comprendre l'intérêt des heuristiques et des métaheuristiques, il faut d'abord s'intéresser à la notion de complexité. Cette dernière représente les ressources nécessaires à un algorithme pour qu'il s'exécute, usuellement le temps de calcul est mesuré. La complexité est segmentée en deux différentes classes de problèmes : P et NP. La classe P regroupe les problèmes qui peuvent être résolus dans un temps polynomial. Nous considèrons les problèmes dans la classe P comme pouvant être résolu efficacement, sinon le problème est dît difficile. Les problèmes constituant la classe NP sont ceux dont il est possible de vérifier que la solution est atteignable dans un temps polynomial. Si P est incluse dans NP, l'inverse n'est pas sûr, et encore à ce jour le problème $P = NP$ n'est pas résolu.
On comprend ainsi l'intérêt de développer des méthodes efficaces, et aussi rapides que possibles, pour résoudre des problèmes difficiles. Les méthodes d'optimisation sont réparties suivant plusieurs familles : mono-objectif vs multi-objectif, continue vs discret, parralélisée ou nons, etc [112]. Les heuristiques et les métaheuristiques font parties de la famille des méthodes stochastiques, c'est-à-dire que l'algorithme ne va pas retourner la même solution entre deux exécutions indépendantes à cause de l'aspect aléatoire présent dans la méthode.

Le mot heuristique a pour origine le grec ancien eurisko qui signifie "trouver" et qualifie tout ce qui utile à la découverte, à l'invention et à la recherche. Les heuristiques sont des méthodes d'optimisation relativement simples et rapides (avec une complexité de temps polynomial) pour résoudre des problèmes difficiles. Ces méthodes sont généralement spécifiques à un type de problème et sont donc dépendantes de celui-ci, contrairement aux métaheuristiques. Le terme *métaheuristique* est composé de méta provenant du grec ancien qui signifie "au-delà" qui est traduisible par "à un plus haut niveau" et de heuristique évoqué plus haut. Ce terme a été mentionné pour la première fois par Fred Glover [47] lors de la conception de la recherche tabou : "La recherche avec tabou peut être vue comme une "métaheuristique", superposée à une autre heuristique. L'approche vise à éviter les optimums locaux par une stratégie d'interdiction (ou, plus généralement, de pénalisation) de certains mouvements.

Comme évoqué dans le paragraphe présentant l'énoncé d'un problème d'optimisation, les fonctions objectifs ont plusieurs optima et l'enjeu pour les métaheuristiques est de ne pas rester bloquées dans un optimum local. Cela peut être évité en veillant au respect de l'équilibre entre exploration (ou diversification) et exploitation (ou intensification). L'exploration décrit la capacité de l'algorithme à parcourir l'ensemble de l'espace de recherche S pour cibler des régions prometteuses. L'exploitation est la capacité d'explorer précisément à une région de l'espace de recherche. Les métaheuristiques doivent donc veiller à alterner l'exploration et l'exploitation afin de se diriger vers l'optimum global sachant que plus elles avancent dans la recherche et plus il va être difficiles pour elles d'explorer.
Dans ce qui suit, nous présentons les prinicipales métaheuristiques à solution unique présentes dans la littérature.

1.3 Métaheuristiques à solution unique

Les métaheuristiques à solution unique ont plusieurs noms : elles peuvent être appelées méthodes de recherche locale ou bien méthodes de trajectoire. Leur mécanisme consiste à faire évoluer itérativement une solution dans l'espace de recherche afin de se diriger vers l'optimum global. Nous présentons les méthodes les plus connues qui sont : la méthode de descente, le recuit simulé, la recherche tabou, et la méthode GRASP.

1.3.1 Algorithme de descente

La méthode de descente peut sembler la plus intuitive et la plus simple à comprendre dans le domaine de l'optimisation. Elle est d'ailleurs appelée *hill climbing* dans les problèmes de maximisation. L'algorithme commence par sélectionner aléatoirement une solution, puis à chaque itération la meilleure solution dans le voisinage de la solution courante est sélectionnée. L'algorithme s'arrête lorsque plus aucune amélioration n'est possible. Pour choisir la meilleure solution dans le voisinage il existe plusieurs stratégies différentes, l'algorithme peut choisir celle qui a la meilleure *fitness* par rapport à toutes les autres solutions dans le voisinage, ou choisir la première solution du voisinage qui améliore la *fitness*, il peut aussi sélectionner la solution qui améliore le moins la *fitness* ("la moins bonne solution"), il est aussi possible de choisir la solution au hasard, etc. La principale faiblesse de cette méthode de descente est qu'elle se trouve facilement piégée dans un optimum local. Une amélioration de cet algorithme consiste à lancer plusieurs redémarrages lorsqu'un optimum local est trouvé, en repartant d'une nouvelle solution générée aléatoirement, il s'agit d'algorithme de descente avec relance (ou *random-restart hill climbing*).

1.3.2 Le recuit simulé

Le recuit simulé, *simulated annealing* (SA), tire son principe de la métallurgie. Il consiste à effectuer des cycles de refroidissement lent et de réchauffage d'un matériau afin de minimiser son énergie, en s'appuyant sur les lois de thermodynamique de Boltzmann. Kirkpatrick, C.D. Gelatt et M.P. Vecchi en 1983, puis V. Černy en 1985, ont adapté cette méthode à l'optimisation : la fonction objectif f correspond à l'énergie du matériau, et les paramètres de l'algorithme sont le critère d'arrêt, la température initiale T_0 (plusieurs méthodes existent [33], le critère de changement de palier de température, et la fonction de décroissance de la température. Lors de la recherche de l'optimum la température diminue, l'algorithme commence par une marche aléatoire, puis les mauvaises solutions sont de moins en moins souvent acceptées, et l'algorithme converge vers une solution de l'espace de recherche. Un compromis est donc à trouver afin d'adapter la décroissance de la température pour éviter une trop forte décroissance de celle-ci et ainsi risque de piéger l'algorithme dans un optimum local. D'ailleurs, il existe plusieurs lois de décroissante de la tem-

pérature [33, 37].
Le recuit simulé a été adapté pour résoudre les problèmes d'optimisation continue par Patrick Siarry [102], de plus il a connu un large essor dans différents domaines d'application.

1.3.3 La recherche tabou

La méthode de recherche tabou a été proposée par Fred Glover en 1986 [47], elle introduit la prise en compte du passé par l'utilisation d'une mémoire des solutions explorées lors de la recherche de l'optimum. Cette mémoire est appelée liste tabou car elle contient les solutions déjà été visitées et par lesquelles il est interdit de repasser. L'objectif de cette mémoire est d'empêcher l'algorithme d'être piégé dans un optimum local car elle l'autorise à passer par des solutions qui détériorent la fitness. La taille de la mémoire, le nombre d'éléments mémorisés, est un paramètre de l'algorithme qui permet de prendre en compte l'équilibre de diversification et d'intensification évoqué précédemment. En effet, si la mémoire est faible alors elle va favoriser l'intensification, car un nombre de solutions restreint seront interdites. Cependant, plus la taille de la mémoire augmente, et plus la diversification est favorisée, car l'algoithme pourra de moins en moins visiter les régions précédentes qui ont de grandes chances d'être voisines à la solution actuelle.
La procédure de cette métaheuristique commence par une première solution initiale qui est générée aléatoirement, puis la méthode va sélectionner itérativement la meilleure solution dans son voisinage et mémoriser la solution précédente. Ceci permet alors d'éviter les phénomènes dîts de cyclage (l'algorithme tourne en boucle sur les mêmes solutions).
Il existe des méthodes ayant une mémoire adaptative [111], c'est-à-dire que la taille de la mémoire va pouvoir s'adapter selon le contexte de la recherche et même créer de nouvelles solutions.

1.3.4 Le GRASP

Un algorithme glouton suit un principe très simple consistant à améliorer la solution en recherchant dans son voisinage une solution meilleure. Il va donc agir localement dans une zone spécifique.
La procédure de recherche gloutonne aléatoire adaptative, plus couramment appelée

méthode GRASP (*Greedy Randomized Adaptive Search Procedure*), a été proposée par Feo et Resende [43]. Cette méthode alterne les phases de construction et d'amélioration jusqu'à atteindre le critère d'arrêt. L'étape de la construction du GRASP est similaire à une méthode gloutonne randomisée, elle génère une solution réalisable issue d'une liste de choix potentiels appelée *restricted candidate list* (RCL). Cette liste est triée, c'est la partie gloutonne de l'algorithme. L'étape d'amélioration utilise la solution générée lors de la phase précédente comme une solution initiale pour effectuer une recherche locale. Cette dernière peut être une descente, une recherche tabou, ou toute autre heuristique.

1.4 Métaheuristiques à population de solutions : intelligence par essaims

Les métaheuristiques à population de solutions, contrairement aux méthodes à solution unique, font évoluer simultanément un ensemble de solutions dans l'espace de recherche. Il y a d'ailleurs souvent une intéraction entre les solutions qu'elle soit directe ou indirecte afin de faire évoluer la population. Deux grandes classes de métaheuristiques à population de solutions sont distinguables : les algorithmes évolutionnaires inspirés de la théorie de l'évolution de Darwin [26], et les algorithmes d'intelligence en essaim inspirés de biologie ou de l'éthologie [12]. Dans ce paragraphe, nous nous intéressons à cette dernière catégorie en présentant quatre méthodes : l'optimisation par colonies de fourmis, l'optimisation par essaims particulaires, l'optimisation par systèmes immunitaires, et l'optimisation par bio-géographie.

1.4.1 Optimisation par colonies de fourmis

L'optimisation par colonies de fourmis, *Ant Colony Optimization* (ACO), conçue par Dorigo [32] s'inspire comme son nom l'indique du comportement des fourmis lorsque celles-ci cherchent de la nourriture et optimisent le chemin entre leur nid et la nourriture trouvée [30]. En effet, les fourmis utilisent leur environnement pour communiquer entres elles, il s'agit d'un mécanisme dît stigmergique grâce auquel elles déposent des phéromones sur le sol pour signifient aux autres fourmis le chemin qu'elles ont parcouru pour atteindre la nourriture. Ainsi, les autres pourront suivre la piste de phéromones pour retrouver la source de nourriture. Or, il se trouve

que les phéromones s'évaporent avec le temps, par conséquent ce sont les chemins les plus courts qui conserveront une concentration de phéromones plus importante. C'est comme cela que les fourmis trouvent naturellement le plus court chemin à leur nourriture depuis leur abris.
Le ACO reprend la notion de système multi-agents dans lequel chaque agent est représenté par une fourmi. Cela peut être par exemple utilisé pour parcourir un graphe : une fourmi parcoure le graphe de manière aléatoire, mais avec une probabilité plus importante de suivre une arête du graphe, en fonction de la quantité de phéromones déposée dessus. Lorsque le graphe est entièrement parcouru, elle laisse sur le chemin qu'elle a pris une quantité de phéromones proportionnelle à la longueur de ce chemin.

1.4.2 Optimisation par essaims particulaires

La méthode des essaims particulaires, *Particle Swarm Optimization* (PSO), a été conçue en 1995 [36]. Le principe de la méthode provient de l'observation des comportements collectifs d'animaux, tels que le déplacement en vol des oiseaux, des insectes, ou des bancs de poissons, elle s'inspire des modélisations statistiques développées par Reynolds [93], Heppner et Grenander [55]. Les solutions sont ici appelées particules, elles représentent des solutions potentielles qui parcourent l'espace de recherche, la population est donc vue comme un essaim particulaire.
Concrètement chaque particule possède une position X_i, une vitesse V_i. Le but va donc être de déplacer les particules vers la meilleure solution possible, pour cela le déplacement de chaque particule va prendre en compte sa position courante vers la meilleure solution qu'elle a trouvée depuis le début de la recherche(notée P_i), et enfin elle va aussi tenir compte de la meilleure solution trouvée par l'essaim entier (notée P_g). Chacun de ces trois aspects sont respectivement appelés des composantes physique, cognitive et sociale. Cela se traduit mathématiquement par :

$$\overrightarrow{V_i(t+1)} = \omega \times \overrightarrow{V_i(t)} + cC1 \times r_1 \times (\overrightarrow{P_i(t)} - \overrightarrow{X_i(t)}) + C_2 \times r_2 \times \overrightarrow{P_g(t)} - \overrightarrow{X_i(t)}) \quad (1.3)$$

avec (C_1, C_2) deux constantes représentant une accélération positive, (r_1, r_2) sont deux nombres aléatoires tirés selon une loi de distribution uniforme dans l'intervalle $[0; 1]$, $i = 1, 2, 3, ..., N$ avec N la taille de l'essaim, et ω est le coefficient inertie [Shi

Eberhart, 1998](Clerc, et al., 2002). La nouvelle position est calculée comme suit :

$$\overrightarrow{X_i(t+1)} = \overrightarrow{X_i(t)} + \overrightarrow{V_i(t+1)} \tag{1.4}$$

Afin de contrôler le pas des particules dans l'espace de recherche et contrôler l'équilibre diversification - intensification, il est possible de borner la vitesse dans un intervalle $[-V_{max}; V_{max}]$ en fixant une vitesse maximale [35].
Cette méthode a connu beaucoup de succès car comme les métaheuristiques précédentes, il est possible de l'hybrider avec d'autres méthodes, d'en créer des variantes [20, 101] (comme l'algorithme Tribes [19, 23]), et de l'appliquer dans de nombreux domaines d'application ([6, 124, 130]).

1.4.3 Optimisation par systèmes immunitaires artificiels

L'optimisation par systèmes immunitaires artificiels, *artificial immune systems* (AIS), est née dans les années 1980 grâce au travail de Farmer, Packard et Perelson [42]. L'AIS mime le fonctionnement du système immunitaire des êtres humains. En effet, ce dernier a pour but de protéger le corps d'agents pathogènes extérieurs comme les bactéries ou les virus. Il est composé de cellules ainsi que d'organes.
Il existe quelques groupes d'AIS (plus en détails [28, 113]) :

1. les réseaux immunitaires artificiels [61] ;
2. les algorithmes de sélection négatifs [45] ;
3. les algorithmes de sélection clonale [29] ;
4. la théorie du danger [2, 17], dont le but est de concevoir des mécanismes de détection d'intrusion pour la sécurité informatique ;
5. les algorithmes de cellules dendritiques [50].

Voir [27] pour plus d'informations plusieurs articles sont disponibles.

1.4.4 Optimisation par bio-géographie

La méthode biogéographique, *biogeography-based optimization* (BBO), conçue par Dan Simon en 2008 [104], provient de la théorie de l'équilibre dynamique énoncée par MacArthur et Wilson [118]. Celle-ci consiste à étudier la répartition et le nombres des espèces vivantes dans à un endroit donné.

On considère une île, et elle est supposée initialement vide, puis que les espèces vont venir petit à petit des îles alentours. Chaque espèce est définie par des capacités spécifiques lui permettant de venir s'installer plus ou moins facilement sur l'île, et elles vont donc rentrer en concurrence pour leur survie. Le système est considéré à l'équilibre lorsque les espèces sont constamment remplacées, donc que le taux d'immigration égal le taux d'émigration.
L'algorithme BBO perçoit chaque solution potentielle comme une île dont la fitness est déterminée par la fonction appelée *habitat suitability index* (HSI) et est représentée par des *suitability index variables* (SIV), un taux d'immigration λ et un taux d'émigration μ. Lorsque l'île est vide le taux d'immigration est maximal, inversement lorsque le taux de capacité maximal S_{max} d'espèces est atteint le taux d'immigration est nul, et inversement pour le taux d'émigration. Le taux d'immigration est défini tel que :

$$\lambda = I \times (1 - \frac{S}{S_{max}}) \tag{1.5}$$

avec I le taux d'immigration maximum, S la capacité actuelle, et S_{max}la capacité maximale d'espèces sur l'île. De même avec le taux d'émigration :

$$\mu = E \times (1 - \frac{S}{S_{max}}) \tag{1.6}$$

avec E le taux d'émigration maximum.
Au début de l'exécution du BBO, le taux d'immigration chute rapidement lorsque les meilleurs espèces s'installent sur l'île. Le taux d'émigration s'accroît plus rapidement avec un nombre élevé d'espèces déjà présentes sur l'île. Le nombre d'espèces à l'équilibre sur l'île (S^*) est déterminé par l'intersection des courbes d'émigration (E) et d'immigration (I). Il existe toutefois différents modèles mathématiques de la biogéographie qui comprennent des variables plus complexes [118].

1.5 Métaheuristiques à population de solutions : algorithmes évolutionnaires

Dans ce paragraphe, nous allons décrire la seconde moitié des métaheuristiques à population de solution : les algorithmes évolutionnaires. Nous commencerons par introduire les différentes notions nécessaires à la compréhension de ces méthodes.

Puis nous détaillerons les algorithmes génétiques, l'optimisation par stratégie d'évolution, les algorithmes à estimation de distribution, l'évolution différentielles, les algorithmes coévolutionnaires, les algorithmes évolutionnaires guidés par la diversité, et pour terminer les algorithmes évolutionnaires adaptatifs.

1.5.1 Introduction et rappels

Les algorithmes évolutionnaires (AE) appartiennent à la famille des algorithmes d'optimisation bioinspirés [13], et plus largement au champ de l'Intelligence Artificielle (bas niveau). Les AE sont basés sur la théorie darwinienne de l'évolution des espèces animale et végétale qui met en avant l'adaptation que celles-ci ont dû effectuer pour survivre aux changements de leur environnement. En 1859, Charles Darwin (1809 - 1882) développe dans son ouvrage *De l'origine des espèces* les idées de la sélection naturelle comme moteur de l'évolution, et l'unité ainsi que la diversité des espèces vivantes s'expliquent par cette évolution, qui seront le fondement des AE.

A propos du principe de sélection naturelle, nous pourrions dire que les caractéristiques augmentant la survie et la reproduction de l'espèce vont voir leur fréquence d'apparation augmenter d'une génération à une autre et permettre l'adaptation de l'espèce. La sélection naturelle repose alors sur trois principes : le principe de variation, le principe d'adaptation, et le principe d'hérédité. Le premier, le principe de variation, établi que les individus diffèrent les uns des autres par leur caractéristiques telles que par la taille, le poids, les traits, la couleur, la pilosité, etc, il s'agit de variation. Le principe d'adaptation signifie que les individus ayant des variations de caractéristiques permettant d'augmenter leur taux de reproduction et de survie, sont alors considérés comme les mieux adaptés à leur environnement, et ils ont donc un avantage sélectif puisque leur progéniture est plus nombreuse. Enfin, le principe d'hérédité repose sur le fait que les variations des caractéristiques avantageuses dans une espèce sont héréditaires, autrement dit transmissible à la génération suivante. D'où la citation devenue célèbre : "Ce n'est pas la plus forte ni la plus intelligente des espèces qui survivra, mais celle qui sera la plus apte à changer".

Retournons à l'optimisation, il faut attendre les années 1950 pour voir le principe d'évolution darwinienne transcrit pour aboutir aux algorithmes évolutionnaires. Des années plus tard, Rechenberg et Schwefel [98] travaillèrent sur les algorithmes de stratégie d'évolution, et Fogel [44] développa aussi la programmation évolutionnaire.

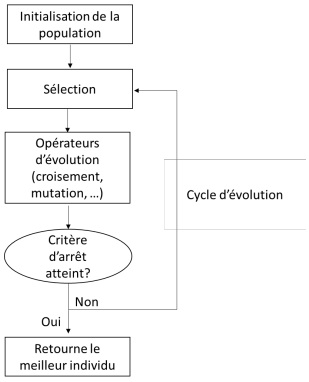


Figure 1.2 – Organigramme d'un algorithme évolutionnaire type.

En 1975, John Holland conçu le premier algorithme génétique [59], il s'agit aujourd'hui de l'un des algorithmes d'optimisation les plus populaires. Ces trois catégories d'algorithmes évolutionnaires ont été dans les années 1990 regroupés sous l'appelation de *Evolutionary Computation*, calcul évolutionnaire. Cependant, l'optimisation évolutionnaire comprend aussi la programmation génétique [64] qui est apparue plus tardivement au début des années 90. Par la suite, d'autres auteurs ont d'hybridé les EAs, il y a pour exemple les algorithmes mémétiques qui sont des EAs couplés à des recherches locales [78]. Chacune de ces hybridations présente une approche différente, et améliore incrémentalement sur les 40 dernières années.

La figure Fig.1.2 illustre le fonctionnemet d'un AE. La première étape consiste à initialiser la population, puis chaque étape correspond à un opérateur d'évolution ayant un pouvoir de modification sur les individus en fonction de l'algorithme évolutionnaire appliqué. Les opérateurs d'évolution les plus connus sont la sélection, le croisement, la mutation et le remplacement. La figure Fig.1.2 met en exergue un cycle d'évolution se répétant afin de produire une nouvelle génération d'individus afin de faire évoluer la population de solutions vers des zones prometteuses, et donc de trouver l'optimum local. Les AE sont des méthodes approchées, leur but n'est donc pas trouver la solution exacte, mais de trouver des solutions satisfaisant au mieux des contraintes parfois contradictoires, et au vu de leur popularité, ils ont d'ailleurs prouvé leur efficacité.

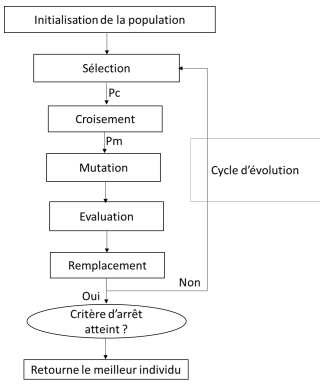


Figure 1.3 – Organigramme d'un algorithme génétique.

1.5.2 Algorithmes génétiques

Les algorithmes génétiques (AG) sont peut-être les algorithmes évolutionnaires les plus populaires, appliqués dans de nombreux domaines tels ques le traitement d'images, l'optimisation de réseaux, l'optimisation de fonction numériques difficiles, la planification, etc. Les AG ont été conçus dès le début des années 1970 grâce à John Holland [59], puis David Goldberg [49] a contribué à leur démocratisation. Le principe consiste à simuler le principe d'évolution biologique évoqué plus haut au niveau des chromosomes des individus.

La figure Fig.1.3 illustre l'organigramme d'un AG : l'initialisation de la population se déroule en premier lieu, (elle est généralement aléatoire, mais des opérateurs ont été développés [8] puis intervient la boucle d'évolution permettant la production de nouvelles générations de population. Les étapes de ce cycle sont : la sélection des individus reproducteurs, ou parents, puis l'application d'une méthode croisement suivant une probabilité $P_c \in [0, 1]$, puis l'application d'une méthode de mutation là aussi suivant une probabilité $P_m \in [0, 1]$, et enfin le remplacement des anciens individus par de nouveaux individus générés dans la population. A noter que les AG ont une taille de population fixe tout au long du processus d'optimisation. La sélection permet de sélectionner les individus parents à la génération n afin de constituer des descendants pour la génération $n + 1$. Il existe pour la sélection, comme pour les autres opérateurs, différentes méthodes possibles afin d'y arriver, par exemple : la sélection aléatoire, la sélection par roulette, la sélection par tournoi, la sélection par

rang, etc, [11, 48].
L'opérateur de croisement effectuer une combinaison des gênes des parents afin de générer la progéniture. Une multitude d'opérateurs de croisement dans la littérature tels que le croisement en n-points, le croisement uniforme, etc, [67]. La Fig.1.3 illustre que l'application du croisement dépend d'une probabilité, appelée taux de croisement, noté $P_c \in [0; 1]$ dont laquelle l'application de l'opérateur sur les parents sélectionnés va dépendre (si $x = rand(0, 1)$ est tel que $x > P_c$ alors les 2 parents sélectionnés ne subiront pas de croisement, et donc les enfants seront les copies des parents, sinon l'opérateur est appliqué et les enfants sont des combinaisons des gênes de parents).
La mutation permet de modifier les gênes des descendants individuellement. La figure Fig.1.3 montre que comme pour l'opérateur de croisement, il existe un taux de mutation, noté $P_m \in [0; 1]$, déterminant la proportion de descendants subissant une mutation. Là encore il y a dans la littérature [67] diverses méthodes la mutation de Levy, la mutation *scramble*, etc. Le remplacement est la dernière étape permettant d'établir une nouvelle génération. La taille de la population étant fixe, l'opérateur de remplacement, peut consister à insérer systématiquement les descendants dans la population à la place des parents, ou de ne conserver que les meilleurs individus parents ou enfants, etc.

1.5.3 Stratégie d'évolution

La stratégie d'évolution (SE) date des années 1960, initialement développée par Rechenberg [91] puis par Schwefel pour la conception de profils aérodynamiques. La SE utilise un ensemble de μ parents pour générer λ enfants. Pour produire chaque enfant, ρ parents se recombinent, puis les enfants sont mutés, généralement en ajoutant une variable aléatoire suivant une loi normale. Il y a deux méthodes de sélection : la première, appelée sélection virgule, consiste à sélection uniquement les enfants (on note la méthode $(\mu/\rho, \lambda)$-ES), dans la seconde, appelée sélection plus, il s'agit de prendre l'ensemble parents et enfants (méthode notée $(\mu/\rho + \lambda)$-ES). A noter qu'il existe des SE hiérarchiques appelées Meta-ES ou *Nested-ED*.
L'algorithme le plus connu de la SE est probablement le CMA-ES (*Covariance Matrix Adaptation Evolution Strategy*) proposée par Hansen [52]. Cet algorithme est défini par l'auto-adaptation de la matrice de covariance de la distribution normale utilisée pour la mutation. Cette métaheuristique est encore aujourd'hui considérée

comme l'une des plus performantes.

1.5.4 Algorithmes à estimation de distribution

Les algorithmes à estimation de distribution (EDA) ont été proposés en 1996 par Mühlenbein et Paab. Bien qu'ils soient inspirés des algorithmes génétiques, les EDA n'ont pas d'opérateurs de croisement ou de mutation. En effet, ils estiment les relations entre les différentes variables du problème grâce à une estimation de la distribution de probabilité de l'échantillon de solutions. Celui-ci varie à chaque itération à partir des paramètres de la distribution, estimés à l'itération précédente. Les EDA sont divisés en trois catégories en fonction de la complexité des modèles probabilistes choisis pour évaluer les dépendances entre variables [66] :

- les modèles sans dépendance (*Univariate EDA*) : la distribution de probabilité est construite à partir d'un ensemble de distributions univariantes, donc indépendantes sur chaque variable. Ces modèles sont simples mais peu représentatifs des problèmes d'optimisation difficile car les dépendances sont souvent nombreuses. Un exemple de problème est le *one max* ;
- les modèles à dépendances bi-variantes (*Bivariate EDA*) prennent en compte deux dépendances du modèle ;
- enfin les modèles à dépendances multi-variantes (*Multivariate EDA*) prennent en compte toutes les dépendances dans le modèle.

Les variantes les plus connues de l'EDA sont l'apprentissage incrémental à population (PBIL) [5], l'algorithme à distribution marginale univariée (UMDA) [73], et l'algorithme génétique compact (cGA) [53], il existe aussi des méthodes d'EDA basées sur le partitionnement de données pour l'optimisation multimodale, sur le calcul parallèle, etc.

1.5.5 Évolution différentielle

L'évolution différentielle, *Differential Evolution* (DE) a été proposé par Storn et Price en 1997 [107] afin de résoudre le problème d'ajustement par polynômes de Tchebychev. Depuis beaucoup de DE différentes ont été développées [89], dont une des plus performantes est le *multi-population ensemble DE* (MPEDE) [120]. Il existe une nomenclature particulière $DE/x/y/z$, avec x le mode de sélection de l'individu

(rand ou best) pour la mutation, y le nombre de vecteurs de différences utilisés pour la perturbation du vecteur cible x, et z l'opérateur de croisement (binomial ou exponentiel). Un algorithme de DE consiste à itérer les étapes suivantes : muter un individu parent $\overrightarrow{P_i}$ afin de générer un nouvel individu $\overrightarrow{Ch_i}$, puis un vecteur d'essai $\overrightarrow{U_i}$ est généré après le croisement entre les individus $\overrightarrow{P_i}$ et son descendeant $\overrightarrow{Ch_i}$, puis la sélection consiste à choisir le meilleur individu entre $\overrightarrow{P_i}$ et $\overrightarrow{Ch_i}$.

1.5.6 Algorithmes coévolutionnaires

Les algorithmes coévolutionnaires (CoEA), ont été élaborés en 1990 par Hillis [56] via ses travaux sur les réseaux de tri. Les CoEA sont basés sur la coévolution biologique des espèces vivantes en partant du principe qu'elles peuvent s'influencer de manière coopérative ou concurrentielle. Concrètement les CoEA gèrent plusieurs populations d'individus en parallèle qui vont évoluer et intéragir les unes avec les autres en coopération ou en concurrence.
Concernant les CoEA coopératives, les individus des populations sont récompensés quand ils fonctionnent bien ensemble et pénalisés quand ils échouent ensemble [88]. Inversement pour les CoEA concurrentielles, les populations sont mises en compétition pour résoudre le problème à optimiser et les individus sont récompensés contrairement à ceux avec lesquels ils intéragissent qui sont pénalisés [105].

1.5.7 Algorithmes évolutionnaires guidés par la diversité

Les algorithmes évolutionnaires guidés par la diversité ont pour but d'améliorer les performances de l'algorithme génétique qui a tendance à être coincé rapidement dans les optima locaux. Ainsi l'AE est observé par le principe de la métrique de diversité, et des mécanismes sont mis en place, conservant la nature intrinsèque de celui-ci, mais en y effectuantdes adaptations via les opérateurs ou leur application. Nous remarquons qu'il s'agit d'algorithmes multi-populations et non-adaptatifs.
Les algorithmes les plus connus sont le *Diversity-Control-Oriented Genetic Algorithm* (DCGA) [100], le *Shifting-Balance Genetic Algorithm* (SBGA) [85], le *Forking Genetic Algorithm* (fGA) [115], et le *Preference-Based Genetic Algorithm* (PBGA) [82].

1.5.8 Les algorithmes évolutionnaires adaptatifs

Les algorithmes évolutionnaires adaptatifs sont une catégorie d'AE capables de s'ajuster eux-mêmes en fonction de grandeurs internes et/ou externes faisant parties du contexte d'exécution. Ils permettent ainsi aux utilisateurs de garantir une simplicité de réglage des paramètres. La clef de l'adaptation est de s'appuyer sur les bonnes métriques et donc de s'intéresser à un critère de performance. Précédemment, nous avons étudié différentes métaheuristiques composant l'état de l'art, et nombre d'entre elles ont été par la suite conçues dans une version adaptative :

- *adaptive PSO* [21, 23, 99, 109, 121, 124, 129],
- *adaptive genetic algorithms* [79, 96, 97],
- *adaptive ant colony optimization* [31, 46],
- *adative tabu search* [7],
- *adaptive DE* [1, 4, 14, 71, 77, 86, 126, 127],
- *adaptive EA* [38],
- *adaptive evolution strategy* [72, 94],
- *adaptive memetic algorithms* [16, 81].

Pour un algorithme "s'adapter" peut signifier plusieurs choses, ou qu'il peut changer les valeurs de certains de ses paramètres, ou appliquer des méthodes différentes, ou sélectionner des composants internes. Cependant, tous les algorithmes cités traitent du réglage automatique de la valeur de leurs paramètres, il ne s'agit pas d'aller modifier dynamiquement des composants internes. Il faut donc être vigilent car derrière une même dénomination commune d' "algorithmes adaptatifs" les notions, et par prolongement les techniques employées sont différentes et variées. Néanmoins, d'autres algorithmes évoqués plus tôt, tels que MPEDE [120], CMA-ES [72], ENA [125], PBGA [82], MS-CAP [60] ou SAHJA [80], ont proposé de coordonner la recherche de différents mécanismes intériorisés de fitness (fitness, diversité, etc).

1.6 Conclusion

Dans ce chapitre, nous avons présenté ce qu'est un problème d'optimisation, puis nous avons distingué les heuristiques des métaheuristiques. Ensuite, nous avons présenté les principales métaheuristiques à solution unique, puis à population de solutions, portant un intérêt particulier pour les algorithmes évolutionnaires.

Nous avons pu observer que la famille des algorithmes évolutionnaires est relativement ancienne et très variée, mais toujours performante. Elle se compose notamment des trois principales catégories : les algorithmes à stratégie d'évolution, la programmation génétique, et les algorithmes génétiques. Cependant, nous avons voulu aussi porter notre attention sur deux autres types d'AE, les algorithmes évolutionnaires guidés par la diversité ainsi que les algorithmes évolutionnaires adaptatifs.

En effet, les premiers essayent de pallier un point faible des algorithmes génétiques qui tendent à converger trop vite (et perdent donc leur diversité) pour se retrouver piéger dans un optimum local. Ces algorithmes gardent néanmois la structure principale d'un AG et ne sont pas adaptatifs. Quant aux AE adaptatifs, ils ne s'adaptent qu'à la marge, c'est-à-dire en effectuant un auto-réglage des paramètres. Nous avons finalement vu que des algorithmes qui ne sont pas qualifés d'adaptatifs proposent de coordonner la recherche grâce à différents mécanismes intériorisés.

Partant de ces observations, nous avons voulu développer la notion d'algorithme évolutionnaire adaptatif afin de proposer des algorithmes dont la conception est plus simple pour l'utilisateur et les performances vérifées. Dans cette thèse, nous présentons deux approches adaptatives pour l'optimisation continue. Cependant cela nous a alors amener à utiliser des méthodes d'apprentissage, dont nous faisons l'état de l'art dans le chapitre suivant.

Chapitre 2

Proposition d'un algorithme évolutionnaire adaptatif basé sur le maximum a posteriori

2.1 Introduction

Dans ce chapitre, nous allons présenter une métaheuristique adaptative basée sur le maximum a posteriori, *Maximum a posteriori based Evolutionary Algorithm* (MEA). Le MEA est conçu pour résoudre des problèmes d'optimisation continue et son processus se termine lorsque la condition d'arrêt est atteinte. Comme tout algorithme évolutionnaire, il utilise une population de solutions qui évoluent au fur et à mesure des générations afin de parcourir l'espace de recherche et trouver la solution optimum du problème. Le MEA suit les cycles classiques de sélection, croisement, mutation, remplacement des individus dans la population. Sa spécificité réside dans l'ajout d'une couche de gestion des opérateurs afin de sélectionner les meilleurs à appliquer au cours de la recherche, et améliorer les performances par rapport à un algorithme évolutionnaire classique.
Le chapitre commence par le rappel théorique du maximum a posteriori (MAP), puis il décrit la structure de la métaheuristique proposée, enfin une analyse des sensibilités, de la complexité, ainsi que des résultats comparatifs sont effectués.

2.2 Principe du maximum a posteriori

Nous allons rappeler les fondements théoriques du maximum a posteriori afin de comprendre son utilisation et son utilité dans le MEA. Le principe du maximum a posteriori (MAP) appartient à la théorie bayésienne et permet d'estimer des valeurs de paramètres. Pour cela il s'appuie sur la distribution a priori contenant l'ensemble

des données passées (dans le cas du MEA, les diversités) d'une (ou de plusieurs) variable(s) (ici les stratégies).
Considérons m échantillons de points de données $y_1, ..., y_m$, chacun de dimension arbitraire. Soit $y = y_1, ..., y_m$ l'ensemble complet des données. Ainsi, y est une variable aléatoire dont la fonction de densité de probabilité est notée $f_y(y_1, ..., ym)$. Une notation standard (en analyse bayésienne) pour les fonctions de densité de probabilité est utilisée, et la fonction de densité de probabilité de la variable aléatoire y est simplement dénotée par $p(y)$.
Supposons que des échantillons ont été générés à partir d'une distribution de probabilité décrite par un paramètre θ (pas nécessairement scalaire). θ est considéré comme une variable aléatoire. La notation abrégée $p\,(y\,|\theta\,)$ pour représenter la famille de fonctions de densité conditionnelle sur y, notée θ.
Une distribution a priori est spécifiée sur θ, notée $p(\theta)$. Cette distribution représente toute connaissance sur la façon dont les échantillons sont générés avant de les observer.
Le but est la fonction de densité conditionnelle sur θ, étant donné les échantillons observés (données), qui est notée $p\,(\theta\,|y\,)$. C'est ce qu'on appelle la distribution postérieure, et elle indique quelle stratégie sera utilisée compte tenu des données observées.
Les probabilités antérieure, postérieure et de vraisemblance sont toutes liées à la règle de Bayes :

$$p\,(\theta\,|y\,) = \frac{p\,(y\,|\theta\,) \times p(\theta)}{p(y)} \tag{2.1}$$

Plutôt que d'estimer la distribution entière $p\,(\theta\,|y\,)$, il suffit parfois de trouver une seule valeur pour θ.
Le MAP est donc une estimation ponctuelle avec une méthode bayésienne où le but est de trouver θ qui maximise la probabilité postérieure, $p\,(\theta\,|y\,)$. La distinction est entre les θ sous lesquels les données sont les plus probables, et les données θ les plus probables.
Comme la fonction de partition $p(y)$ respecte constamment θ, elle ne sera pas évaluée. Encore une fois, il est généralement plus pratique de travailler avec le logarithme :

$$\hat{\theta}_{MAP} \in \arg\max_{\theta} p(\theta\,|y) = \arg\max_{\theta} p(y\,|\theta)p(\theta) = \arg\max_{\theta} \left(\log p(y\,|\theta) + \log p(\theta)\right) \tag{2.2}$$

Il est évident que les choix raisonnables sont ceux qui n'assignent pas de probabilité nulle à la vraie valeur de θ.
La motivation derrière le choix du MAP est basée sur l'idée qu'il permet de faire face à des événements rares. L'estimateur MAP permettrait à l'algorithme d'incorporer la connaissance préalable qu'il existe une certaine probabilité d'un événement rare, qu'il n'a pas encore vu.
Le préalable est obtenu en utilisant l'expérience pour faire une supposition éclairée de ce que les paramètres devraient être, $\mu_p = [\mu_{p(1)}, ..., \mu_{p(m)}]$ ainsi que leur confiance, qui pourrait être traduite en un écart type $\sigma_p = [\sigma_{p(1)}, ...\sigma_{p(m)}]$.
La fonction antérieure est définie comme suit :

$$p\left(y\,|\theta\right) = \prod_{n=1}^{m} f\left(\frac{y - \mu_p}{\sigma_p}\right) \tag{2.3}$$

Nous rappelons que $p\left(y\,|\theta\right)$ dans eq.(2.3) n'est valide que si les probabilités sont mutuellement indépendantes.
Le MAP est la règle sur laquelle est basé le MEA, son utilisation dans la sélection des stratégies de recherche est présentée dans le paragraphe suivant.

2.3 Description de l'algorithme

Afin d'avoir une vue d'ensemble de cette nouvelle approche, le MEA est présenté comme une architecture construite selon trois couches de calcul, comme décrit par la figure 2.1. Le premier, en orange situé en bas de l'organigramme, représente le problème à résoudre. Ce dernier communique avec l'algorithme évolutionnaire (AE) en assignant à chaque individu une *fitness*. L'AE est donc au niveau supérieur, en bleu foncé sur la figure et effectue la recherche dans le domaine de définition, via les opérateurs d'évolution (sélection, croisement, mutation, remplacement) afin de trouver l'individu atteignant la meilleure solution du problème. Le MEA fonctionne comme n'importe quel AE classique entre chaque intervalle de Δ générations. La contribution du MEA se trouve dans l'ajout d'un niveau supplémentaire, prenant en considération les diversités produites lors des dernières générations Δ par la population de l'AE au niveau inférieur. C'est cette couche qui va effectuer la simulation de populations des diverses stratégies. L'avantage des simulations basées sur la diversité est qu'elles n'augmentent pas le nombre d'évaluations alors qu'elles permettent

Algorithme 1 : Algorithme évolutionnaire basé sur le Maximum a posteriori.

Initialisation : Population initialisée aléatoirement, de taille P et de dimension D
Phase d'entraînement :
- Exécution sur la population initiale pendant Δ générations de chaque stratégie
- Calcule la probabilité a posteriori pour chaque stratégie
- Sélectionne la stratégie à appliquer

Phase d'évolution :
tant que : le critère d'arrêt n'est pas atteint
Exécute la stratégie sélectionnée pendant Δ générations :
- Sélectionne les individus qui vont devenir parents ;
- Applique le croisement sur les parents sélectionnés selon le taux de croisement ;
- Applique la mutation sur les parents sélectionnés selon le taux de mutation ;

Applique les stratégies non-sélectionnées sur Δ générations générées sur les populations passées issues de la stratégie sélectionnée
Mesure des diversités
Met à jour la probabilité a posteriori de chaque stratégie
Sélectionne la prochaine stratégie à appliquer
Résultat : La meilleure solution

de comparer les opérateurs entre eux. Cette dernière couche retourne les meilleurs opérateurs à appliquer sur la population pour l'intervalle de prochaines Δ générations.

Ces trois niveaux d'abstraction s'articulent lors du fonctionnement du MEA qui se découpe en 3 grandes étapes. Ces dernières sont décrites dans les paragraphes suivants, ainsi que dans Algorithme 1. La première étape consiste à initialiser aléatoirement la population de solutions, puis elle s'enchaîne avec une phase d'entraînement qui consiste à sélectionner pour la première fois la stratégie à appliquer, puis l'algorithme se termine par la phase d'évolution composée d'une boucle d'instructions à effectuer tant que la (ou les) condition(s) soi(en)t atteinte(s). L'évolution est découpée par intervalle de Δ générations afin d'adapter dynamiquement les opérateurs à appliquer au fur et à mesure de la recherche.

Il est à signaler que la seconde est ce qui distingue le MEA d'un AE classique puisque les opérateurs auraient été choisis par l'utilisateur.

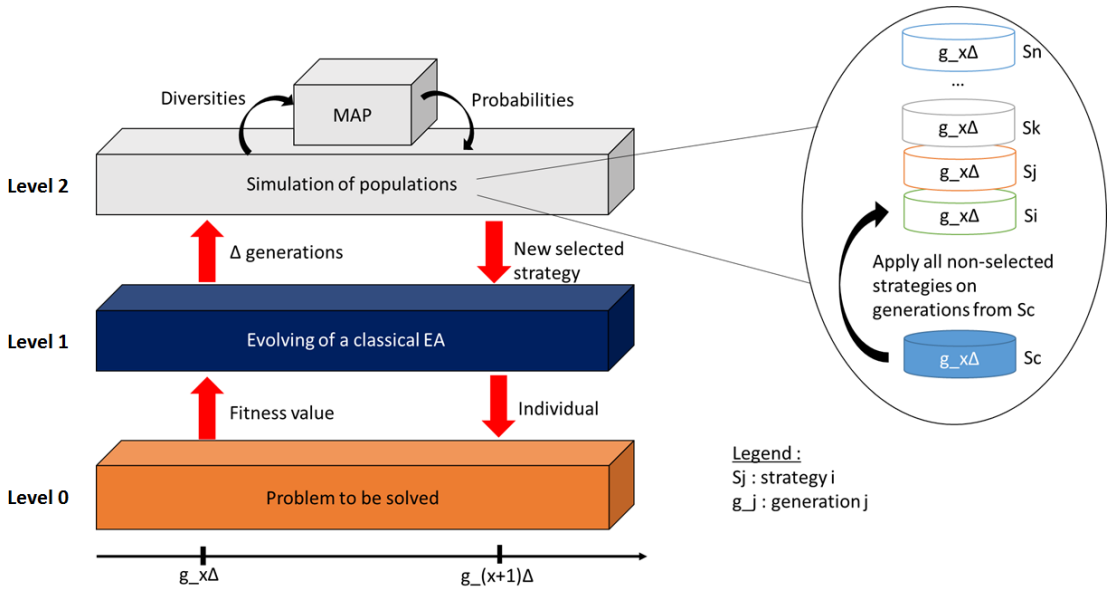


Figure 2.1 – Organigramme de l'approche proposée pendant la phase d'évolution. Le niveau 2 est spécifique à l'approche. Il permet de simuler des populations que des stratégies non sélectionnées auraient produites si elles avaient été choisies et ainsi de changer la stratégie appliquée S_c au niveau 1 à chaque génération Δ.

2.3.1 Initialisation

En raison de la grande diversité des populations initiales, les AEs sont naturellement des algorithmes explorant l'espace de recherche. La détermination de la population initiale n'est pas considérée dans ce travail, néanmoins cette étape a un rôle dans l'efficacité de l'algorithme et sera d'explorer dans les travaux futurs. Dans le MEA, l'initialisation, première ligne de l'algorithme 1, est effectuée aléatoirement (selon une loi uniforme) dans l'espace de recherche.

2.3.2 Phase d'entraînement

Après l'initialisation, on se retrouve avec d'une part les stratégies évolutionnaires et d'autre part la population initialisée. Il s'agit alors de déterminer la meilleure stratégie à appliquer pour la première fois, d'où le besoin d'une phase d'entraînement qui va permettre de départager les stratégies entre elles au début de la recherche. Ainsi lors cette deuxième phase, illustrée par la figure 2.2, les N stratégies de re-

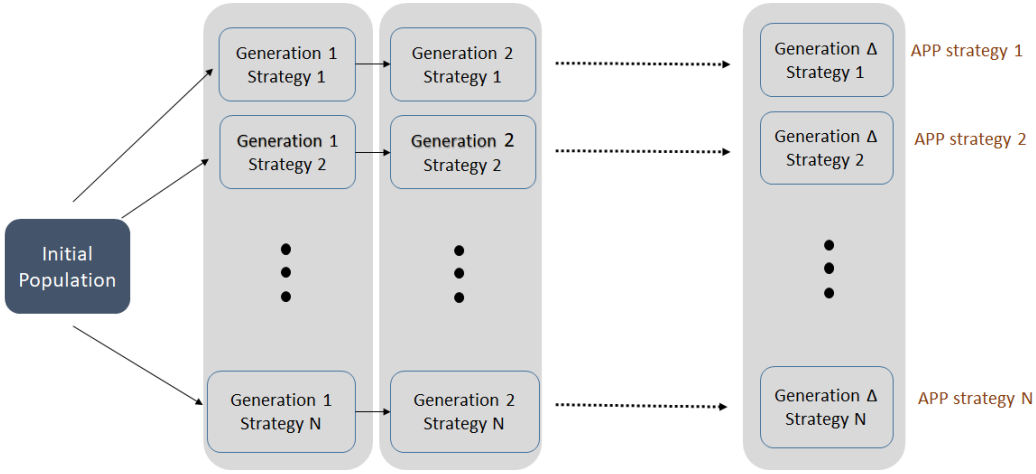


Figure 2.2 – Illustration de la phase d'entraînement avec N stratégies. Chaque stratégie est appliquée parallèlement durant Δ générations à partir de la même population initiale. Note : APP est l'abréviation de probabilité a posteriori.

cherche sont appliquées en parallèle pendant Δ générations à partir de la même population issue de la phase d'initilisation précédente. A chaque génération, la diversité de chaque stratégie est sauvegardée, puis les N diversités sont converties en probabilités a posteriori afin d'utiliser le principe du maximum a posteriori sur celles-ci. En effet, la stratégie S_c à appliquer est la stratégie maximisant la probabilité future d'une grande diversité pour les prochaines générations Δ, en se basant sur les Δ générations passées.

2.3.3 Phase d'évolution

Pour expliquer le déroulement de cette étape, nous allons nous appuyer sur un exemple en prenant en compte un ensemble de $N = 4$ stratégies de recherche, voir l'illustration figure 3.4. On suppose que la stratégie sélectionnée précédemment est $S_c = 1$. Le MEA commence avec l'application de S_c pendant les générations Δ. Ensuite, séparément pour chaque génération, les 3 autres stratégies sont appliquées aux générations issues précédemment de S_c. Ceci, afin de simuler ce que pourrait être la population de chaque stratégie qui n'a pas été sélectionnée aupravant. Puis, avec la même méthodologie employée lors de la phase d'apprentissage, les diversités sont mesurées, les probabilités a posteriori sont calculées et le MAP est utilisé pour sélectionner la stratégie à appliquer pour les prochaines Δ générations sur la population. La différence avec la phase précédente d'apprentissage réside dans la si-

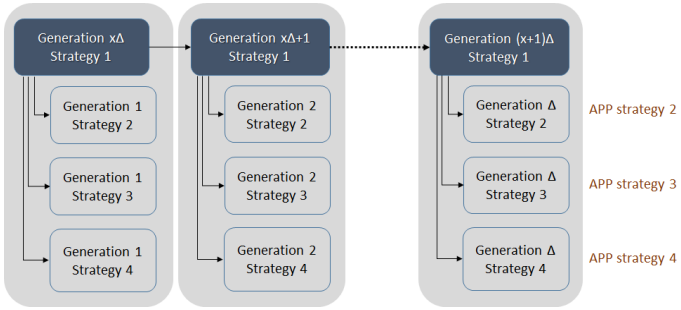


Figure 2.3 – Exemple avec $N = 4$ stratégies du processus de simulation lors de la phase d'évolution, correspondant au niveau 2 de l'organigramme. La stratégie appliquée pendant Δ générations à la population ici est $S_c = 1$. Ensuite, les autres stratégies sont appliquées séparément sur la population issue de S_c à chaque génération. On visualise ainsi la différence avec le processus illustré figure 2.2.

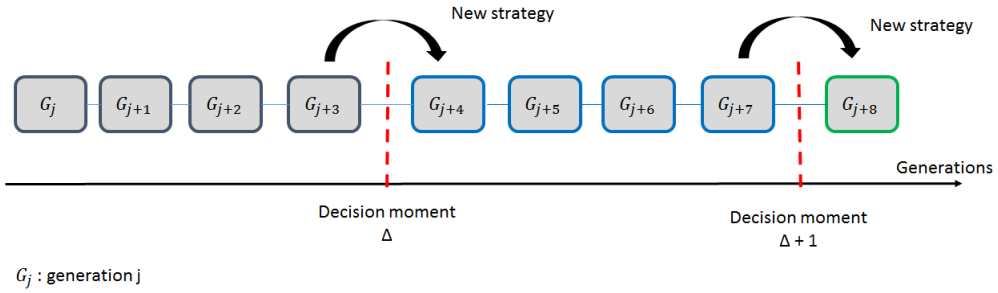


Figure 2.4 – Illustration macroscopique de la population lors de la phase d'évolution correspondant au niveau 1 de l'organigramme. La recherche de l'optimum est effectivement réalisée par une seule population et la stratégie appliquée est le seul composant qui change à interval régulier.

mulation des générations car toutes les stratégies ne sont pas appliquées en parallèle à partir d'une même population initiale, mais chaque génération i simulée est issue de celle S_c à la génération i.

Si l'on se réfère à la figure 2.1, ce dernier niveau permet de favoriser l'exploration et ainsi d'optimiser la fonction objectif en évitant d'avoir une population piégée dans un optimum local. De plus, cela réduit l'effort de conception en recherchant à la place de l'utilisateur les meilleurs opérateurs à appliquer. Enfin, ce mécanisme peut être aussi considéré comme une méthode adaptative visant à maximiser la diversité future, en utilisant l'information des diversités passées.

2.3.4 Mesure de performance

Les métaheuristiques oscillent entre les phases d'exploration et d'exploitation, aussi appelées diversification et intensification respectivement. Pour rappel, l'intensification est la recherche d'une solution dans une zone restreinte de l'espace, alors que la diversification caractérise la recherche globale de solution à travers celui-ci. La performance des AEs dépend de leur capacité à parcourir l'espace de recherche, ainsi leur capacité à passer d'une phase d'exploration à une phase d'exploitation va être un critère déterminant pour atteindre l'optimum.
Pour savoir si un AE intensifie ou explore, la grandeur utilisée est la diversité. Celle-ci va mesurer la similitude entre les individus d'une population (plus ils se ressemblent, plus ils sont proches dans l'espace et plus la diversité va être diminuer).
Les métriques permettant de mesurer la diversité sont issues du domaine mathématique (distance euclidienne, distance de Hamming) ou physique (inertie, entropie). Les notations sont les suivantes : G un gène, P la taille de la population, D la dimension, i le rang du gène et j et j' les rangs des chromosomes. Pour commencer, la distance euclidienne, utilisée par le MEA, est définie comme suit :

$$Dist_{euc} = \sqrt{\sum_{j=1}^{P-1} \sum_{j'=j+1}^{P} \sum_{i=1}^{D} (G_{ij} - G_{ij'})^2} \tag{2.4}$$

La définition de la distance de Hamming est la suivante :

$$Dist_{ham} = \sum_{j=1}^{P-1} \sum_{j'=j+1}^{P} (\sum_{i=1}^{D} \mid G_{ij} - G_{ij'} \mid) \tag{2.5}$$

L'inertie mesure la distribution de masse de la population, la population est imaginée comme une masse physique composée de solutions. Elle est définie par :

$$c_i = \frac{\sum_{j=1}^{P} G_{ij}}{P} \tag{2.6}$$

$$I = \sum_{i=1}^{D} \sum_{j=1}^{P} (G_{ij} - c_i)^2 \tag{2.7}$$

avec c_i le centroïd.
L'entropie est une notion thermodynamique permettant de déterminer la quantité de désordre dans un ensemble de particules, on peut ainsi faire l'analgie avec les

Notation	Définition
D	Taille de l'individu ou dimension
P	Taille de la population
P_c	Taux de croisement
P_m	Taux de mutation
Δ	Intervalle de générations entre chaque changement de stratégie
N	Nombre de stratégies
FEs	Nombre maximum d'évaluations

Tableau 2.1 – Sommaire des paramètres du MEA.

individus formant la population et évoluant au sein de l'espace de recherche.

$$E(P) = -\sum_{k} p_k \times log(p_k) \tag{2.8}$$

avec p_k la proportion de la population qui a la *fitness* de la k^{th} partition. Néanmoins, l'entropie nécessite des évaluations de la fonction objectif, or un critère commun des AEs est d'économiser les évaluations, cette métrique n'a pas donc pas été retenue.

2.4 Résultats et discussions

Toutes les expérimentations sont effectuées sur un ordinateur Intel (R) Core (TM) i7-37703.40 GHz et 8-GB RAM sur Windows 7. Tous les paramètres MEA sont résumés dans le tableau 2.1. Tout d'abord, le paramétrage de l'algorithme est détaillé, puis les variations de la diversité sont analysées, l'utilité du MAP plutôt qu'un choix aléatoire de la stratégie est prouvée. Ensuite, les différentes sensibilités des paramètres sont étudiées au cas par cas. Enfin, une comparaison de l'efficacité de MEA avec d'autres algorithmes est discutée, basée sur un ensemble de 34 problèmes et la complexité basée sur l'évolution du temps d'exécution est détaillée.

2.4.1 Paramétrage de l'algorithme

Pour effectuer nos expérimentations, le MEA est constitué d'un ensemble de stratégies de recherche composé d'opérateurs de croisement et de mutation. Les méthodes de sélection et de remplacement n'ont pas été prises en compte dans la composition des stratégies, mais elles pourraient l'être dans des travaux futurs. De nombreux opérateurs ont été proposés dans la littérature, ici cinq méthodes de croise-

ments et quatre méthodes de mutations parmi les plus connues ont été sélectionnées. Ainsi le MEA a 5 méthodes de croisement et 4 de mutation, soit 20 stratégies à sa disposition. La composition des stratégies est décrite dans le tableau 2.2, le détail de chaque opérateur est donné dans la suite.

Les enfants sont représentés par la notation suivante Ch_1=($G'_{1,1}$,..., $G'_{1,N}$) et Ch_2=($G'_{2,1}$,..., $G'_{2,N}$) et les parents par P_1=($G_{1,1}$,..., $G_{1,N}$) et P_2=($G_{2,1}$,..., $G_{2,N}$), avec N la dimension.

Croisement BLX-α [41] : deux enfants sont générés par deux parents en choisissant aléatoirement pour chaque gène un nombre contenant dans l'intervalle [b_{inf}, b_{sup}]. Ici $\alpha = 0,3$.

$c_{max} = \max(G_{1,N}, G_{2,N})$, $c_{min} = \min(G_{1,N}, G_{2,N})$

$I = c_{max} - c_{min}$

$b_{inf} = c_{min} - I \times \alpha$, $b_{sup} = c_{max} + I \times \alpha$

$G'_{1,k} = rand(b_{inf}, b_{sup})$, $G'_{2,k} = rand(b_{inf}, b_{sup})$ avec $k \in \{1; N\}$

Croisement discret : cet opérateur génère de nouveaux individus en sélectionnant les gènes des parents selon une distribution uniforme.

$G'_{1.k} = set(G_{1,k}, G_{2,k})$

$G'_{2.k} = set(G_{1,k}, G_{2,k})$ avec $k \in \{1; N\}$

Croisement linéaire [119] : trois enfants sont générés en utilisant les différentes opérations linéaires suivantes :

$G'_{1,k} = 0.5 \times G_{1,k} + 0.5 \times G_{2,k}$

$G'_{2,k} = 1.5 \times G_{1,k} - 0.5 \times G_{2,k}$

$G'_{3,k} = -0.5 \times G_{1,k} + 1.5 \times G_{2,k}$ avec $k \in \{1; N\}$

Croisement *one-point* [87] : Un point est choisi aléatoirement, noté a, entre 1 et N, avec N la dimension. Puis, les gènes sont échangés entre deux parents avant le point a. Exemple ien dimension 5 avec $a = 4$:

P_1=($G_{1,1}$, $G_{1,2}$, $G_{1,3}$, $G_{1,4}$, $G_{1,5}$)

P_2=($G_{2,1}$, $G_{2,2}$, $G_{2,3}$, $G_{2,4}$, $G_{2,5}$)

Ch_1=($G_{1,1}$, $G_{1,2}$, $G_{2,3}$, $G_{2,4}$, $G_{2,5}$)

Ch_2=($G_{2,1}$, $G_{2,2}$, $G_{1,3}$, $G_{1,4}$, $G_{1,5}$)

Croisement barycentrique : deux enfants sont créés à partir des deux parents à partir du paramètre constant $\alpha \in [0, 1]$. Cette méthode est aussi appelée croisement arithmétique.

$G'_{1,k} = \alpha \times G_{1,k} + (1 - \alpha) \times G_{2,k}$

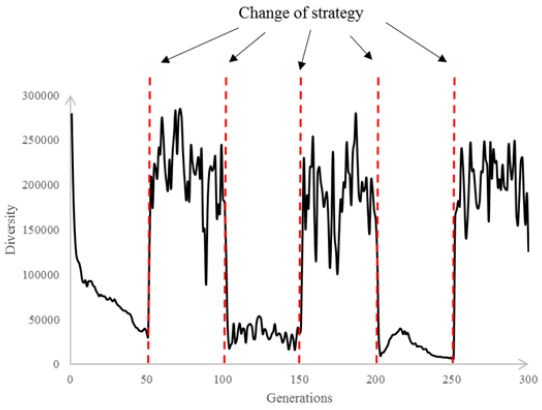


Figure 2.5 – Illustration des variations de la diversité de la population sur la fonction Rastrigin en 100 dimensions sur 300 générations.

$G'_{2,k} = (1 - \alpha) \times G_{1,k} + \alpha \times G_{2,k}$ avec $k \in \{1; N\}$

Mutation de Levy [68, 74] : un nombre tiré aléatoirement, suivant une loi de distribution de Levy et ajouté au gène parent. Pour chaque dimension :

$G'_{1,k} = G_{1,k} + \alpha$, avec $k \in \{1; N\}$

Mutation Gaussienne [57] : elle génère les gènes enfants en perturbant les gènes parents. Cette perturbation suit une loi de distribution uniforme.

$G'_{1,k} = G_{1,k} + g$, avec $k \in \{1; N\}$

Mutation *scramble* [67] : on crée un enfant en échangeant la location des gènes des parents aléatoirement. Exemple en 5 dimensions :

$P_1 = (G_1 G_2 G_3 G_4 G_5 G_6)$

$Ch_1 = (G_4 G_1 G_2 G_6 G_3 G_5)$

Mutation DE/RAND/1/BIN [108] : pour cette méthode, il faut 3 individus pour pouvoir générer un seul enfant. Le nouvel individu est composé d'un gène du premier parent ainsi que la différence entre les gènes des deux autres parents par combinaison linéaire.

$G'_{1,i} = G_{1,i} + F \times (G_{2,i} - G_{3,i})$, avec $F = 0.8$ et $i \in \{1; N\}$

2.4.2 Intérêt du MAP pour le maintien de la diversité

Dans cette expérience, la fonction objectif considérée est Rastrigin en 100 dimensions, la taille de la population est de 50 individus et $\Delta = 50$ générations. La figure 2.5 montre l'évolution de la mesure de la diversité sur 300 générations. Un gradient significatif après chaque intervalle de Δ générations est observé ce qui permet de

Stratégie	Croisement	Mutation
1	BLX-α	Gaussien
2	BLX-α	DE_RAND_1_BIN
3	BLX-α	Scramble
4	BLX-α	Levy
5	Discret	Gaussien
6	Discret	DE_RAND_1_BIN
7	Discret	Scramble
8	Discret	Levy
9	Barycentrique	Gaussien
10	Barycentrique	DE_RAND_1_BIN
11	Barycentrique	Scramble
12	Barycentrique	Levy
13	One-Point	Gaussien
14	One-Point	DE_RAND_1_BIN
15	One-Point	Scramble
16	One-Point	Levy
17	Linéaire	Gaussien
18	Linéaire	DE_RAND_1_BIN
19	Linéaire	Scramble
20	Linéaire	Levy

Tableau 2.2 – Composition des stratégies de recherche.

mettre en évidence l'impact d'un changement de stratégie sur la diversité de la population. A l'intérieur d'un intervalle de Δ générations on observe que la diversité varie aussi, cette fois en fonction de la distribution de la population dans l'espace de recherche. On peut voir que dans le premier intervalle de Δ générations, l'algorithme était coincé dans un optimum local (décroissance rapide de la diversité), puis après changement de la stratégie la diversité augmente. On en déduit que chaque stratégie a son propre potentiel exploitation-exploration. Le MEA fournit une solution au problème de la perte de diversité en utilisant cette propriété observée.
Nous allons maintenant montrer expérimentalement l'intérêt de l'utilisation du MAP. Pour cela, la fonction objectif considérée est Rastrigin en 50 dimensions, la taille de la population est égale à 50 individus et $\Delta = 50$ générations. La condition d'arrêt est le nombre maximal de générations, avec $Max_g = 5000$. Les tests ont été lancés 100 fois et des moyennes sont établies. Une comparaison entre le cas aléatoire (EA, REA) et le cas MAP est présentée dans la figure 2.6 avec différentes valeurs de Δ.

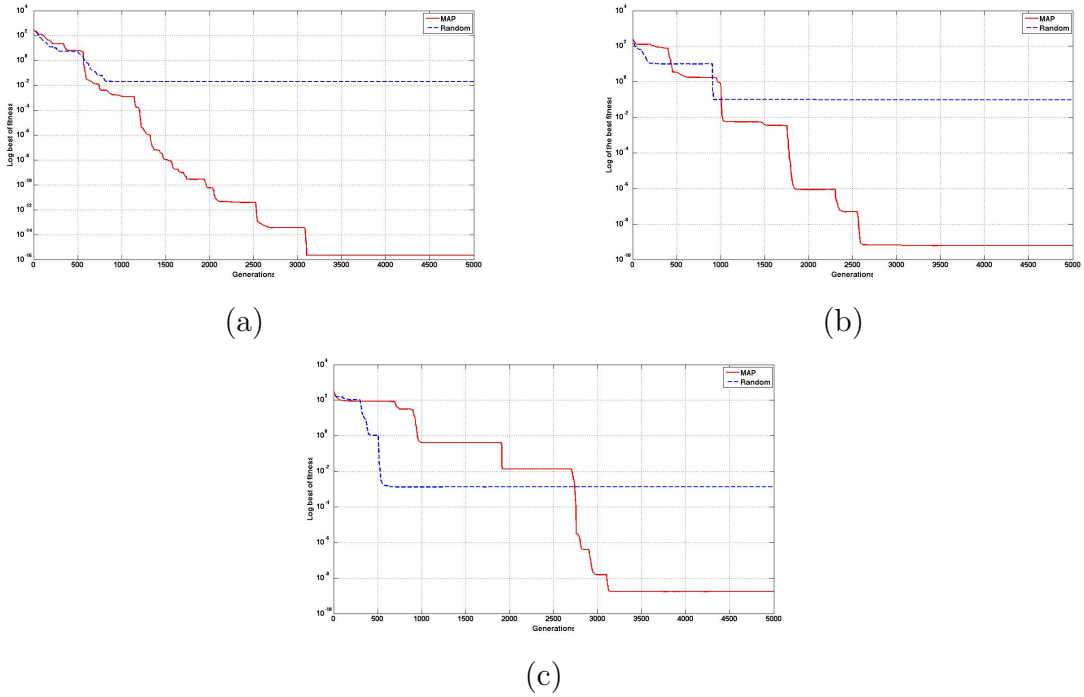


Figure 2.6 – Evolution sur 5000 générations de la meilleure solution (échelle logarithmique) pour MEA et REA lorsque $\Delta = 20$ générations (a), $\Delta = 50$ générations (b) et $\Delta = 100$ générations (c).

Ces résultats soulignent l'intérêt de l'algorithme basé sur le MAP plutôt que de choisir la stratégie au hasard pour obtenir de meilleurs résultats finaux.

2.4.3 Analyse des sensibilités

Sensibilité du paramètre Δ

Dans cette partie, la sensibilité du paramètre Δ est étudiée, ainsi deux cas sont considérés : Δ statique et Δ dynamique.

Δ statique dans la première expérience, l'objectif est d'évaluer les taux de réussite pour différentes valeurs de Δ. Le taux de réussite donne une meilleure compréhension de la vitesse de convergence, il correspond au nombre de fois que les résultats finaux sont inférieurs à $10E{-}08$ après 5000 générations. L'expérience a été réalisée sur le problème de Rastrigin, en 50 dimensions, avec une taille de population de 50 individus. L'évolution de la meilleure solution avec des $\Delta = 10, 20, 50, 100, 200, 500$

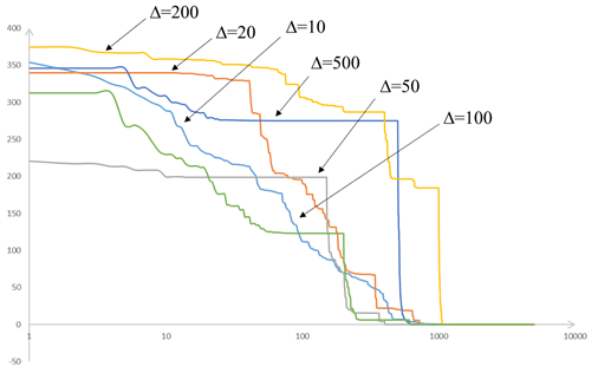


Figure 2.7 – Évolution (à l'échelle logarithmique) de la meilleure solution obtenue par le MEA lorsque les $\Delta = 10, 20, 50, 100, 200$ et 500 générations. Résultats obtenus sur 25 lancements, le nombre de générations maximum est de 5000.

générations est illustrée dans la figure 2.7 et les résultats finaux sont illustrés dans la figure 2.8. Ils montrent que le taux de réussite est élevé (0.99%) lorsque Δ est inférieur ou égal à 50 générations et qu'il diminue pour $\Delta = 100$, 200 et 500 générations.

Dans la seconde expérience, l'objectif est de montrer l'impact des variations Δ sur les performances du MEA. Les fonctions objectif considérées sont Rastrigin et Griewank, elles ont été choisies parce qu'elles ont des propriétés mathématiques différentes, ce qui peut aider à mieux étudier la sensibilité du paramètre Δ. Le critère d'arrêt est le nombre maximum de générations, ici fixé à 5000. Les résultats obtenus sont des moyennes calculées sur 100 lancements, ils sont décris des tableaux 4.2 à 4.11.
En 50 dimensions, d'après les tableaux 4.2 et 4.7, on observe que Δ n'est pas sensible à la dimension du problème et que les meilleurs résultats sont obtenus pour Δ égal à 20 générations. De plus, si l'on considère le problème de Griewank en dimension 100, le résultat final obtenu pour $\Delta = 20$ est proche de la meilleure solution pour $\Delta = 100$ générations. Ainsi, on peut déduire que Δ ne dépend ni de la dimension, ni de la fonction objectif.
La même expérience a été effectuée en augmentant la taille de la population à 100 individus. Les tableaux 4.3 et 4.8 (en Annexes) montrent là encore que les meilleurs résultats sont obtenus avec les générations $\Delta = 20$.
Les figures 2.13 (a) et (b) montrent la distribution des valeurs Δ obtenant les meilleures solutions pour Rastrigin et Griewank respectivement. En ce qui concerne

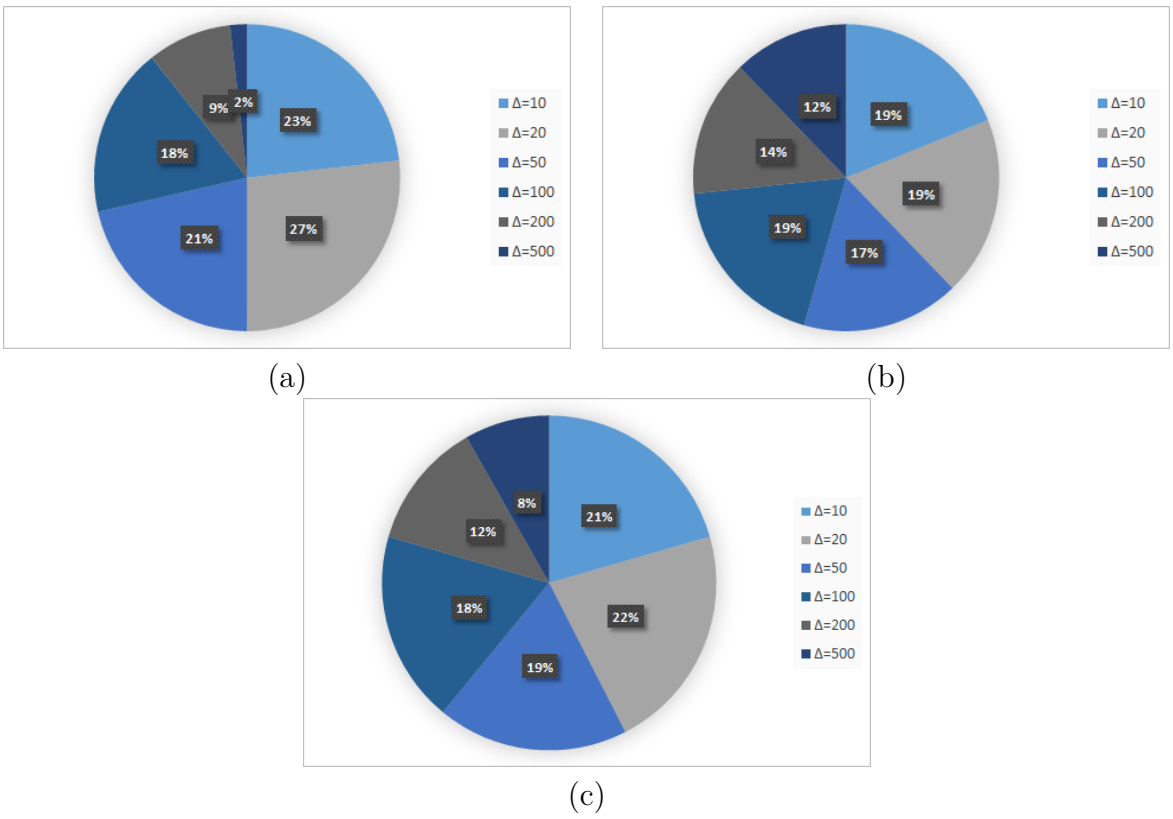


Figure 2.8 – Distribution des meilleures solutions par Δ, les problèmes Rastrigin(a) et Griewank (b) sont testés.

le premier graphique (a), la valeur de Δ la plus efficace est celle égale à 20 générations avec un score allant jusqu'à 27% des résultats. Ensuite, en deuxième position, $\Delta = 10$ obtient 25% et $\Delta = 50$ obtient 21% pour la troisième place. En ce qui concerne le second graphique (b), nous obtenons le même trio mais également réparti, avec 19% de résultats obtenus. Si nous devions choisir une valeur Δ, ce serait 20 puisqu'elle obtient la plus grande distribution pour les deux problèmes (22%) devant $\Delta = 10$ et $\Delta = 50$ avec 21% et 19% respectivement, ce qui conforte l'étude de sensibilité de Δ effectuée précédemment.
On peut ainsi conclure qu'il existe une valeur de Δ optimale et qu'elle est de 20 générations, qu'elle ne dépend ni de la dimension, ni du problème, ni de la taille de la population.

Δ dynamique Cette expérience vise à comparer les résultats finaux obtenus précédemment avec un Δ statique à ceux obtenus avec un résultat dynamique. Le

Dimension	20	50	100	200
Exp	$1.74E{-}05$	**8.45E-08**	**7.28E-05**	**1.23E-06**
	$(1.73E{-}04)$	**(8.38E-07)**	**(7.25E-04)**	$(6.57E{-}06)$
Lin	$9.99E{-}03$	$6.40E{-}04$	$7.75E{-}03$	$1.05E{+}00$
	$(9.89E{-}02)$	$(4.26E{-}03)$	$(6.99E{-}02)$	$(6.93E{+}00)$
Log	**2.61E-08**	$5.35E{-}03$	$1.29E{-}01$	$1.18E{-}01$
	(1.90E-07)	$(4.78E{-}02)$	$(1.04E{-}14)$	$(9.05E{-}01)$

Tableau 2.3 – Résultats finaux selon Δ pour la fonction Rastrigin et une population de 50 individus.

Dimension	20	50	100	200
Exp	**9.96E-12**	**3.55E-11**	**3.17E-10**	**3.39E-07**
	(8.07E-11)	**(2.54E-10)**	**(1.71E-09)**	**(1.27E-06)**
Lin	$1.61E{-}09$	$3.13E{-}08$	$5.94E{-}08$	$1.41E{-}03$
	$(1.45E{-}08)$	$(2.52E{-}07)$	$(3.61E{-}07)$	$(3.07E{-}03)$
Log	$1.08E{-}10$	$6.23E{-}09$	$1.66E{-}09$	$3.47E{-}07$
	$(8.96E{-}10)$	$(3.87E{-}08)$	$(1.01E{-}08)$	$(1.02E{-}06)$

Tableau 2.4 – Résultats finaux selon Δ pour la fonction Griewank et une population de 50 individus.

Dimension	20	50	100	200
Exp	**6.33E-11**	**0.00E+00**	**3.57E-15**	**3.35E-08**
	(6.26E-10)	**(0.00E+00)**	**(3.55E-14)**	**(1.95E-07)**
Lin	$2.39E{-}07$	$5.13E{-}04$	$1.20E{-}04$	$1.04E{-}05$
	$(1.84E{-}06)$	$(3.61E{-}03)$	$(1.17E{-}03)$	$(4.40E{-}05)$
Log	$8.08E{-}05$	$2.83E{-}04$	$6.00E{-}03$	$1.45E{-}02$
	$(7.51E{-}04)$	$(2.82E{-}03)$	$(5.94E{-}02)$	$(1.41E{-}01)$

Tableau 2.5 – Résultats finaux selon Δ pour la fonction Rastrigin et une population de 100 individus.

problème de Rastrigin est considéré dans les dimensions 50 et la condition d'arrêt est fixée à 5000 générations. Les résultats finaux sont calculés sur 100 lancements et présentés des tableaux 2.3 à 2.6. Trois lois décroissantes différentes sont considérées : exponentielle, logarithmique et linéaire. Ainsi, les intervalles entre les moments de décision évoluent en fonction de ces lois pendant l'exécution, voir la figure 2.9.
La loi décroissante logarithmique obtient les meilleurs résultats par rapport aux lois exponentielle et linéaire. Néanmoins, en comparaison aux résultats finaux précédents, $\Delta = 20$ générations est plus efficace que celui qui suit la loi du logarithme décroissant, donc un Δ statique est préférable à un Δ dynamique.

Dimension	20	50	100	200
Exp	**0.00E+00**	**2.33E-13**	**6.20E-15**	**5.25E-09**
	(0.00E+00)	**(1.88E-12)**	**(4.61E-14)**	**1.67E-08**
Lin	$2.22E{-}12$	$8.27E{-}10$	$6.31E{-}08$	$8.02E{-}07$
	$(1.50E{-}11)$	$(7.93E{-}09)$	$(3.70E{-}07)$	$(2.19E{-}06)$
Log	$1.04E{-}12$	$5.84E{-}12$	$4.83E{-}14$	$2.46E{-}06$
	$(8.17E{-}12)$	$(4.10E{-}11)$	$(3.48E{-}13)$	$(9.81E{-}06)$

Tableau 2.6 – Résultats finaux selon Δ pour la fonction Griewank et une population de 100 individus.

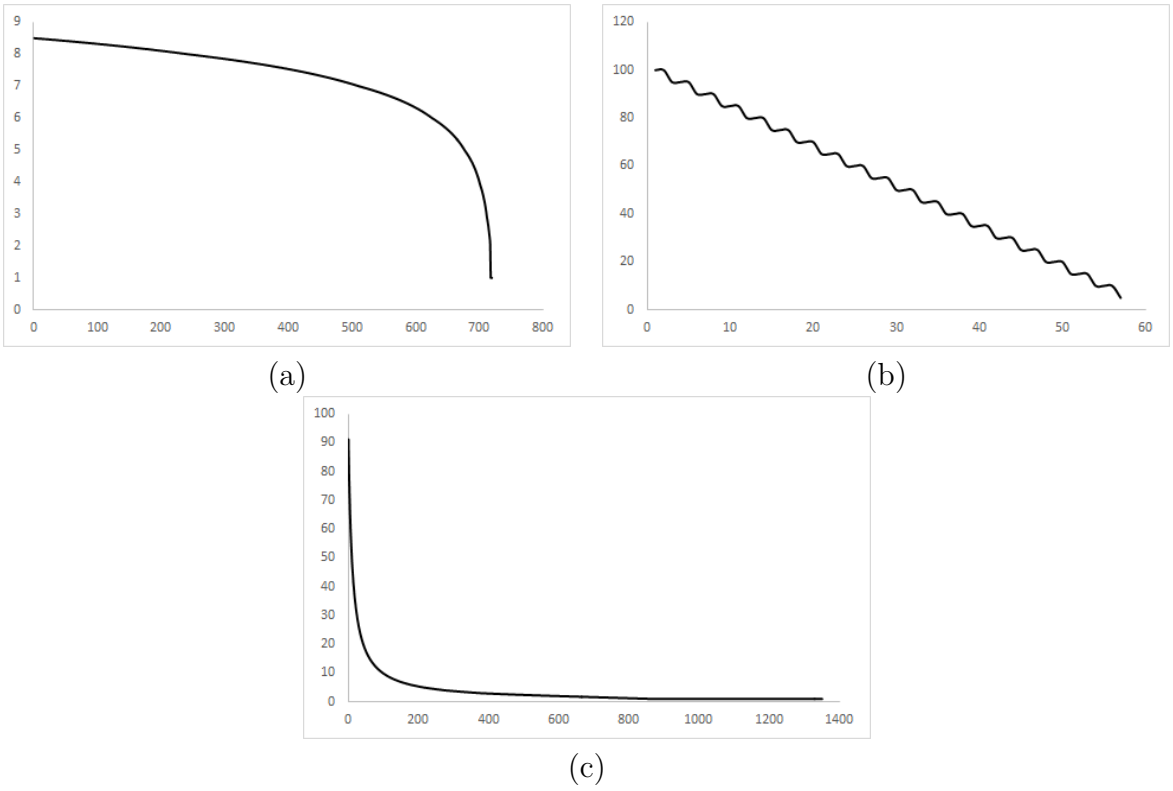


Figure 2.9 – Evolutions de Δ suivant une loi logarithmique décroissante (a), une loi linéaire décroissante (b) et une loi exponentielle décroissante (c). La valeur initiale de Δ est de 100 générations.

Sensibilité du nombre de stratégies

MEA est testé sur les fonctions Sphère et Rosenbrock en dimensions 20 et 50 et les paramètres ont les mêmes valeurs qu'auparavant, la condition d'arrêt est fixée à 5000 générations. Les résultats détaillent les solutions finales moyennes sur 100 lancements et ils sont présentés dans le tableau 2.7. Pour rappel le tableau 2.2 résume la composition des stratégies.

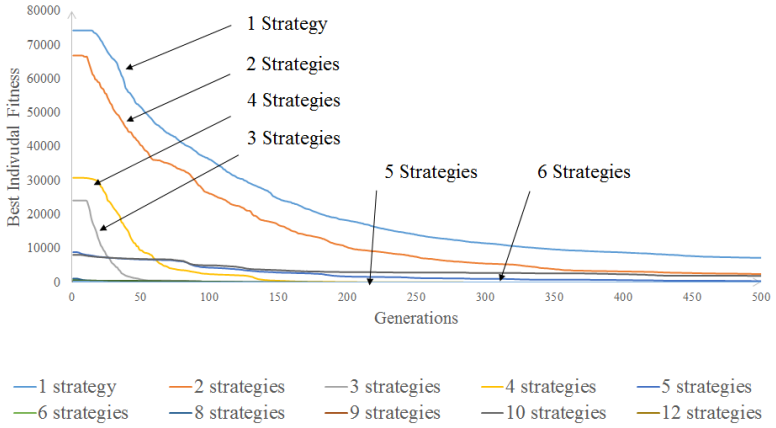


Figure 2.10 – Résultats des tests de sensibilité du paramètre N pour la fonction Sphère.

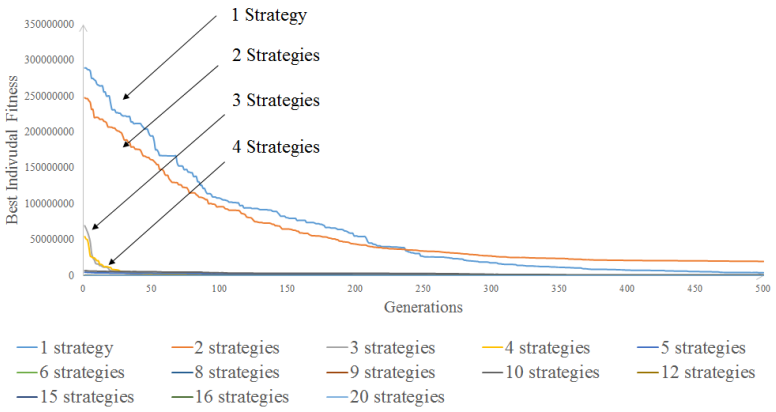


Figure 2.11 – Résultats des tests de sensibilité du paramètre N pour la fonction Rosenbrock.

Les figures 2.10 et 2.11 illustrent l'évolution de la meilleure solution en fonction du nombre de stratégies disponibles N. Les résultats montrent que plus le MEA a des stratégies disponibles, plus la *fitness* est petite dès la première génération. De plus, plus N augmente, plus le gradient est faible sur les 400 premières générations. Néanmoins, la solution finale n'est pas meilleure si le nombre de stratégies disponibles augmente. Comme indiqué dans le tableau 2.7, la meilleure solution finale est obtenue pour 6 stratégies pour la fonction Sphère et pour 15 stratégies concernant la fonction Rosenbrock. La même expérience est effectuée sur le problème Rosenbrock en dimension 20 et cette fois, c'est l'ensemble N composé des 20 stratégies qui obtient la meilleure solution finale.
Pour résumer, l'évolution de la *fitness* du meilleur individu est plus rapide avec un

N stratégies	Sphère 50D	Rosenbrock 50D	Rosenbrock 20D
1	$6.01E{+}3$	$3.44E{+}04$	$2.48E{+}02$
2	$2.01E{+}3$	$1.82E{+}07$	$3.78E{+}02$
3	$4.97E{-}12$	$1.37E{+}01$	$1.50E{+}01$
4	$2.70E{-}01$	$7.32E{+}01$	$8.36E{+}00$
5	$1.80E{+}01$	$7.39E{+}03$	$1.81E{+}02$
6	**4.48E-43**	$7.59E{+}01$	$7.17E{+}00$
8	$1.67E{-}24$	$1.79E{+}01$	$1.14E{+}01$
9	$1.95E{-}19$	$6.03E{-}04$	$8.25E{+}00$
10	$6.56E{+}01$	$3.22E{+}04$	$4.00E{+}02$
12	$4.59E{-}15$	$0.04E{+}00$	$6.72E{+}00$
15	$8.55E{-}23$	**1.82E-06**	$3,81E{+}00$
16	$7.91E{-}07$	$1.04E{+}01$	$7.69E{+}00$
20	$2.66E{-}19$	$4.5E{-}03$	**3.13E+00**

Tableau 2.7 – Les résultats finaux moyens sur 100 lancements (les meilleurs en gras) pour les problèmes Sphère et Rosenbrock en 50 dimensions et Rosenbrock en 20 dimensions et le nombre de générations maximum égal à 5000.

nombre de stratégies N grand, cependant la solution finale n'est pas améliorée en augmentant N. On en déduit donc que la valeur optimale du paramètre N est sensible non seulement au problème, mais aussi à la dimension. On peut alors supposer qu'il existe un compromis à trouver entre la valeur de N et celles des autres paramètres, en mettant ainsi en avant l'hypothèse d'un nombre de stratégies optimal existant.

Sensibilité de la taille de la population

Le but est d'étudier la sensibilité du MEA par rapport à la taille de la population. Pour ce faire, l'expérimentation est réalisée pour des valeurs de taille de population égales à 50, 100, 300, 500 et 800 individus.
L'expérience est réalisée sur des problèmes Rastrigin et Griewank (car ils ont des propriétés différentes) et les résultats sont obtenus pour Δ=10, 20, 50, 100, 200, 500 et dimension D=20, 50, 100, 200. Les autres paramètres sont le taux de croisement $P_c = 0.3$, le taux de mutation $P_m = 0.7$ et la taille de l'ensemble de recherche $N = 20$. Le critère d'arrêt est fixé à 5000 générations, si la solution finale est inférieure à $10E{-}8$ alors elle est considérée comme égale à 0. Les tableaux 4.2 à 4.11

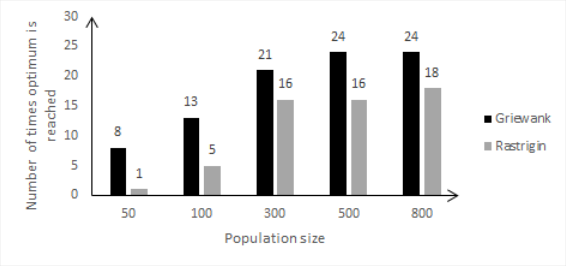
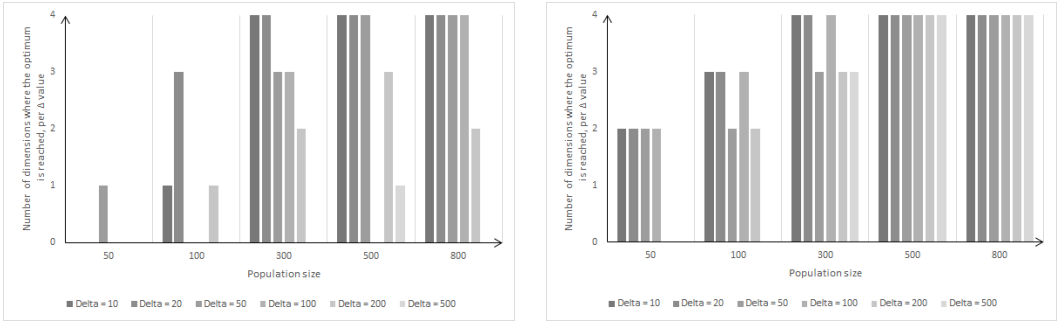


Figure 2.12 – Evolution du taux de réussite en fonction de la taille de la population. En gris, les résultats obtenus pour la fonction Rastrigin problem, en noir ceux obtenus pour le problème de Griewank.



(a) (b)

Figure 2.13 – Evolution du nombre de fois que le MEA atteint l'optimum pour les problèmes Rastrigin (a) et Griewank (b). Quatre dimensions sont testées : $20, 50, 100$ et 200.

détaillent les résultats moyens finaux et les figures 2.12 et 2.13 les illustrent.
Le premier point observé lorsque l'on étudie la figure 2.12 est que pour les deux problèmes, plus la taille de la population augmente et plus l'optimum va être atteint, donc le taux de réussite est meilleur. Ce comportement est prévisible car avec une taille de population plus grande, un AE a plus de chance d'atteindre l'optimum rapidement.
Concernant le problème Rastrigin, les tableaux 4.2 à 4.6 (en Annexes) montrent que $\Delta = 500$ n'atteint jamais l'optimum quelle que soit la dimension, la valeur Δ ou la taille de la population. Puis, à partir de 300 individus, la dimension optimale semble être de 200, ce qui fait que le MEA est plus efficace en grande dimension avec une population importante, là aussi c'est un comporte classique dans les EAs.
Un autre point observé est les valeurs de Δ atteignant la meilleure solution d'une taille de population à une autre : plus N augmente et plus Δ peut être grand et

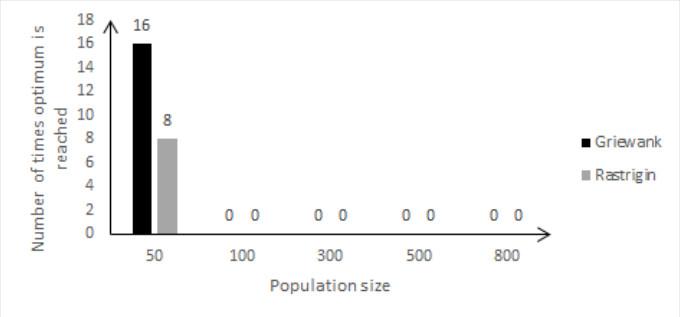
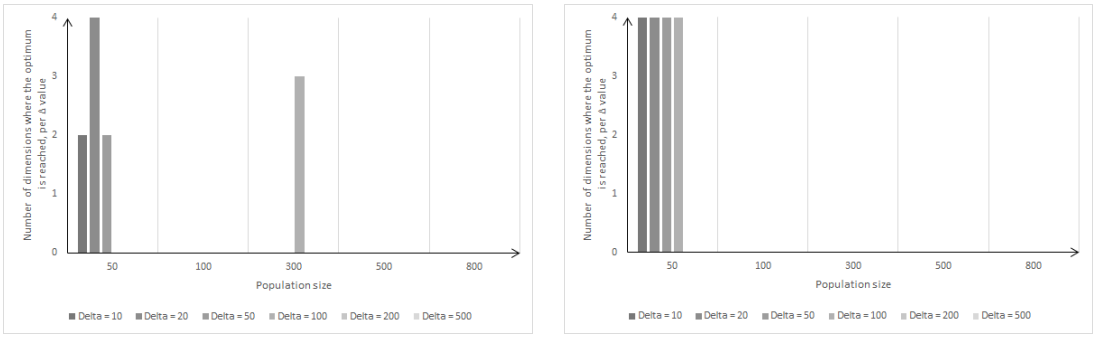


Figure 2.14 – Evolutions du nombre de fois que l'optimum est atteint en fonction de la taille de la population avec des taux de croisement et de mutation égaux à 1. En gris, les résultats proviennent du problème de Rastrigin, en noir ceux du problème de Griewank.



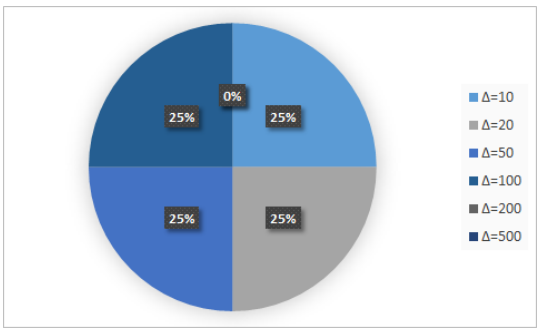
(a) (b)

Figure 2.15 – Considérant le problème de Rastrigin, l'évolution du nombre de fois que chaque Δ atteint l'optimum en fonction de la taille de la population. Quatre dimensions sont testées : $20, 50, 100, 200$ avec des taux de croisement et de mutation égaux à 1.

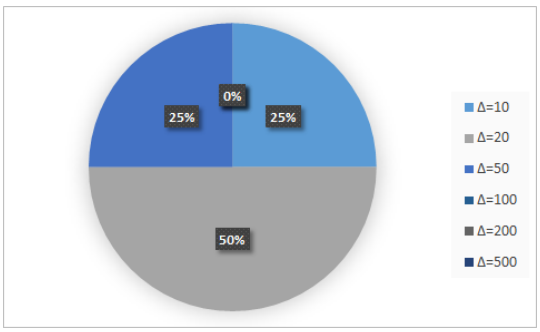
converger. Les mêmes observations peuvent être faites en ce qui concerne le problème de Griewank, voir les tableaux 4.7 à 4.11, excepté pour les générations $\Delta = 500$.
En conclusion, pour résoudre les problèmes d'optimisation de grande dimension, il est préférable d'augmenter la population et qu'en conséquence Δ peut aussi être augmenté sans empêcher la convergence de l'algorithme.

Sensibilité des taux de croisement et de mutation

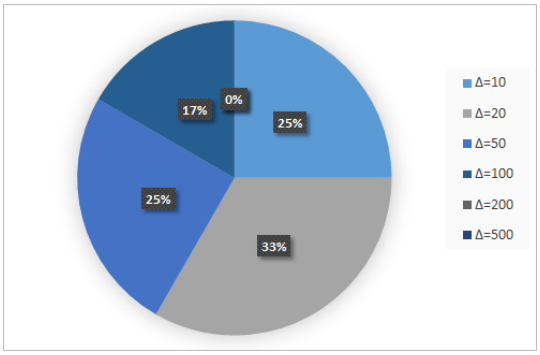
Pour cette expérience, nous avons effectué les mêmes tests que précédemment, avec le même paramètrage, à l'exception des valeurs pour les taux de mutation et de croisement qui ont été mis à 1. Cela signifie que le MEA effectue un croisement et une



(a)



(b)



(c)

Figure 2.16 – Considérant les problèmes de Griewank (a) de Rastrigin (b), , la distribution des meilleures solutions par Δ avec des taux de croisement et de mutation égaux à 1 est illustrée. Le graphe (c) représente la distribution pour les deux problèmes (a) et (b).

mutation systématiquement d'une génération à l'autre. Les illustrations graphiques sont des figures 2.14 à 2.16.

On observe globalement que les résultats ont été clairement impactés, en effet le MEA appliquant le croisement et la mutation de manière systématique réduit ses performances. La figure 2.14 montre que le MEA n'atteint l'optimum que pour une population $P = 50$, à la fois pour les problèmes de Rastrigin et de Griewank. Ensuite, les figures 2.15 confirment que seule une population de 50 individus est capable d'atteindre l'optimum pour $\Delta = 10, 20$ et 50, contrairement aux autres tailles de population. En comparant les Δ des figures 2.15 à 2.16, la diminution de la performance rend le choix d'un paramètre Δ bien distribué moins évident. La figure 2.16 présente $\Delta = 20$ comme le plus efficace (50% de distribution) devant $\Delta = 10$ et $\Delta = 50$ avec 25%. La figure 2.16 montre une égalité entre Δ=10, 20, 50, 100 avec 25%. Enfin, la figure 2.16 (c) résume les figures précédentes, avec la

répartition des meilleures valeurs de Δ : $\Delta = 20(33\%)$, puis $\Delta = 50(25\%)$, enfin $\Delta = 10(25\%)$. On peut remarquer ainsi que les mêmes valeurs de Δ que celles précédemment étudiées atteignent l'optimum, cependant les résultats sont moins satisfaisants avec des taux de croisement et de mutation fixés à 1.

2.4.4 Conclusion

Les résultats montrent que Δ ne dépend pas de la dimension, de la taille de la population ou de la fonction objectif et le nombre de générations 20 semble être une bonne valeur à définir. Néanmoins, le paramètre du nombre de stratégies N est sensible au réglage d'autres paramètres. De plus, le MEA obtient les meilleurs taux de réussite sur un ensemble de problèmes représentant une grande variété de fonctions d'optimisation, ce qui démontre sa robustesse en comparaison aux autres algorithmes, les résultats de Bonferroni-Dunn conforte cette conclusion.

2.5 Conclusion

Une nouvelle métaheuristique a été proposée et détaillée dans ce chapitre. Ce nouvel algorithme, appelé *Maximum a posterior Evolutionary Algorithm* (MEA), est capable de s'adapter tout au long du processus en choisissant la meilleure stratégie de recherche à appliquer. La stratégie choisie est celle qui, d'après le principe du maximum a posteriori (MAP), maximise la diversité future en s'appuyant sur les valeurs passées. En plus des paramètres habituels des algorithmes évolutionnaires, il y a deux paramères spécifiques au MEA : l'intervalle entre les changements d'opérateurs nommé Δ et le nombre de stratégies N disponibles.
Les expérimentations effectuées mettent d'une part en avant les différentes sensibilités des paramètres.
Le chapitre suivant met en avant le second algorithme qui se fonde sur le même principe d'adaptation dynamique des opérateurs appliqués à la population, mais cette fois via l'utilisation d'un graphe de stratégies.

Chapitre 3

Proposition d'un algorithme évolutionnaire adaptatif basé sur un graphe d'opérateurs

3.1 Introduction

Dans ce chapitre, nous allons présenter une métaheuristique adaptative s'appuyant sur un graphe d'opérateurs, appelé *Evolutionary Algorithm based on an Operators Graph* (EAOG). EAOG est conçu afin de résoudre des problèmes d'optimisation continue. L'idée principale est de conserver le principe d'adaptabilité d'un algorithme évolutionnaire, en s'appuyant sur un graphe de stratégies d'évolution. Ce dernier constitue le coeur de notre seconde contribution. Il permet de représenter le passage d'une stratégie à une autre en se basant sur les probabilités passées de diverstié. Pour calculer ces dernières, une méthode d'apprentissage est employée afin de pouvoir établir un modèle de la fonction objectif utilisé lors de simulations. La méthode choisie est un réseau de neurones récurrent bien connu, basé sur la notion de mémoire et appelé réseau de neurones LMSTM (*Long Short-Term Memory*).
Ce chapitre est découpé en trois parties, d'abord le réseau de neurones employé est décrit, puis la méthode proposée est détaillée, et enfin l'analyse des sensibilités des paramètres est présentée.

3.2 Réseau de neurones LSTM (*Long Short-Term Memory*)

Ces dernières décennies, les réseaux neuronaux, ou réseaux de neurones artificiels (RNA), se sont popularisés car ils ont prouvé leur efficacité dans des domaines divers

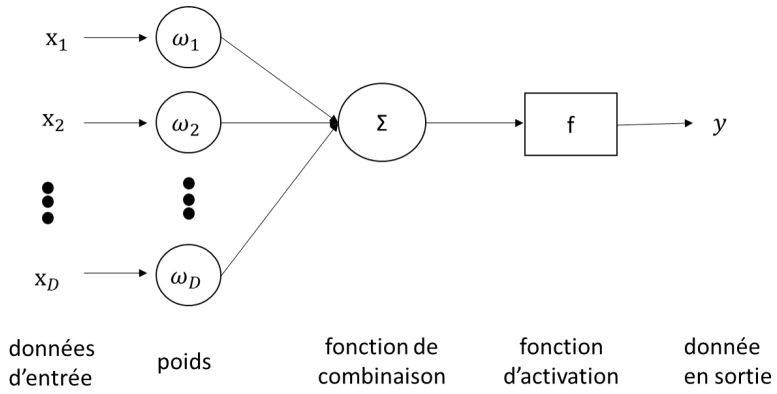


Figure 3.1 – Schéma neurone formel. 5 zones sont définies : les données d'entrée à gauche, sont associées aux poids, la fonction de combinaison effetue une somme pondérée, puis la fonction d'activation s'applique, et enfin y est obtenu en sortie.

tels que le traitement d'images, la simulation d'apprentissage, la robotique, les statistiques, la médecine, l'industrie, la physique, etc. Initialement, les RNA sont basés sur le fonctionnement des neurones présents dans le cerveau humain. Leur apparition remonte aux débuts des années 1940 grâce à McCulloch et Pitts qui ont défini le premier neurone formel [69, 75]. Depuis, la littérature s'est considérablement développée notamment autour des contributions suivantes : [9, 10, 24, 34, 76, 92, 114].

Le neurone biologique reçoit des impulsions électriques par l'intermédiaire des dendrites, puis il renvoie l'information via les axones. La connexion entre dendrites et axone est assurée par les synapses. De plus, les impulsions électriques sont des signaux ayant un effet de seuil sur le neurone. Ces propriétés biologiques sont traduites dans la définition du neurone formel. En effet, un neurone formel est une unité élémentaire de calcul, qui représente un modèle. Le neurone est composé des poids, d'une fonction de combinaison, et d'une fonction d'activation, tout ceci appliqué aux données d'entrée du neurone afin d'obtenir une valeur de sortie, voir la figure 3.1. Il existe différentes fonctions d'activation dans la littérature, les plus connues étant :

— la fonction à seuil,
— la fonction à rampe,
— la fonction sigmoïde,
— la fonction tangente hyperbolique (tanh),

— la fonction de Rectification Linéaire (ReLU) [22],
— la fonction de Rectification Linéaire Paramétrique (PReLU) [54].

Ainsi, un réseau de neurones artificiels est une structure de type graphe, plus ou moins complexe, composée de noeuds, les neurones, pouvant être reliés entre eux, ce qui forme un réseau. Il existe alors diverses catégories de réseaux de neurones, celles-ci dépendent des caractéristiques qui ont été définies :

— l'organisation du graphe : réseau en couches, réseau complet, etc.
— la complexité : nombre de neurones et présence de boucles de rétroaction
— la fonction d'activation du neurone

Nous allons nous intéresser plus particulièrement aux réseaux de neurones récurrents, dont sont issus les réseaux de neurones LSTM. Un réseau de neurones récurrents introduit une boucle de rétroaction dans son architecture, lui permettant de transporter l'information dans les deux sens dans toutes les couches avec un gain constant de 1 (phénomène de disparition du gradient à éviter). Ainsi ils peuvent prendre en compte un nombre d'états passés à chaque instant. Les connexions récurrentes permettent de garder en mémoire des informations bien plus longtemps, ce qui est plus efficaces lors d'applications vidéo à traiter par exemple, de la traduction séquence par séquence, ou encore pour de l'apprentissage lorsque les données forment une suite et ne sont pas indépendantes les unes des autres. De plus, les réseaux de neurones récurrents sont dîts Turing-complets, contrairement aux réseaux de neurones classiques, c'est-à-dire qu'ils possèdent un potentiel plus élevé pour simuler des modèles ou des algorithmes car il a une puissance équivalente aux machines de Turing [103].

Ce type de réseau de neurones a été initialement conçu par Hochreiter et Schmidhuber en 1997 [58], ils font aujourd'hui partie des types de RNA les plus utilisés. Ils sont pour but de contrôler la mémoire via des *gates*. En effet, chaque neurone est composé d'un *input gate*, d'un *output gate* et d'un *forget gate*. L'*input gate* permet d'autoriser ou de bloquer la mise à jour de l'état, de même pour *output gate* pour la sortie du neurone, enfin la *forget gate* permet la remise à zéro de l'état.

Afin d'implémenter le réseau LSTM, nous avons utilisé une librairie open source permettant l'implémentation de méthodes d'apprentissage. Elle a été développée par Google, et est adaptée notamment aux langages Python et C++.

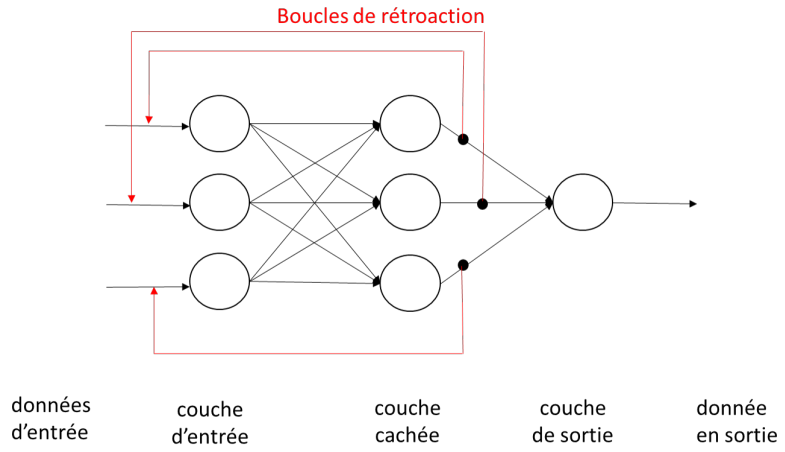


Figure 3.2 – Exemple d'un réseau de neurones récurrent avec 3 dans la couche d'entrée, 3 neurones dans la couche cachée, et 1 neurone dans la couche de sortie.

3.3 Description de l'approche proposée

Afin d'introduire le nouvel algorithme, EAOG, le modèle d'architecture présentant les différentes couches de calcul de la métaheuristique est repris, voir la figure 3.3. Nous retrouvons trois couches réparties comme précédemment, c'est-à-dire que tout en bas, en orange, se situe le problème à résoudre. Au dessus, au deuxième niveau, en bleu foncé, se trouve l'AE qui traite avec le problème en communiquant les individus afin d'obtenir leur fitness, tout en effectuant la recherche via les opérateurs d'évolution mis à sa disposition. Enfin, au niveau supérieur, en gris, se situe la dernière couche de calcul, constituée d'un graphe de stratégies d'évolution qui utilise un réseau de neurones LSTM, et qui constitue la contribution de ce chapitre. Cette dernière couche de gestion prend en entrée les diversités des populations simulées, et retourne la stratégie d'évolution à appliquer sur la population. Le choix de cette stratégie est alors remise en question à intervalle régulier de Δ générations. Les populations sont simulées grâce à l'utilisation d'une méthode d'apprentissage, plus précisément d'un réseau de neurones LSTM, permettant de créer un modèle de la fonction objectif et ainsi de simuler les populations sur un intervalle de Δ générations.

L'architecture est mise en oeuvre lors du fonctionnement de EAOG se déroule en 2 parties : l'initialisation, puis l'évolution qui s'arrête lorsque le ou les critères d'arrêt

Algorithme 2 : AE adaptatif basé sur un graphe de stratégies d'évolution

N stratégies, P_m taux de mutation, P_c taux de croisement
Initialisation :
Population initialisée aléatoirement de taille P et de dimension D
Construction du graphe :
-Crée un graphe de taille N noeuds
-Initialise les poids des arcs à $\omega_{S_iS_j} = \frac{1}{N}$
-Sélectionne aléatoirement la stratégie à appliquer
Construction du modèle : Entraînement du réseau de neurones à plusieurs couches
Evolution : tant que : Critère d'arrêt n'est pas atteint
Appliquer la stratégie sélectionnée pendant Δ générations :
-Sélectionne les individus qui deviendront les parents
-Applique le croisement sur les parents (en fonction de P_c)
-Applique la mutation mutation sur les parents (en fonction de P_m)
Simule les générations des stratégies non sélectionnées sur Δ générations à partir de la population en utilisant le modèle
Mesure les diversités
Met à jour les poids correspondants dans le graphe
Sélectionne la nouvelle stratégie à appliquer
Résultat : La meilleure solution

sont atteints. Ces étapes sont alors détaillées dans Algorithme 2.

Chaque noeud du graphe représente une stratégie d'évolution mise à la disposition de l'algorithme. Le but va etre de mettre à jour les poids du graphe afin de représenter au mieux la perte de diversité en passant d'une stratégie à une autre, afin d'assurer la recherche de la meilleure solution possible dans l'espace de recherche. Pour mettre à jour chaque poids, noté $\omega_{S_iS_j}$ avec S_iS_j l'arc orienté du noeud S_i vers le noeud S_j, l'algorithme va alors simuler les populations issues des stratégies non appliquées pendant l'intervalle de Δ générations précédent. Pour économiser des évaluations de fonction objectif, la solution choisie est donc de se baser sur un modèle approximatif de la fonction objectif.

Le réseau de neurones va apprendre tout au long de la recherche, car l'ensemble des solutions parcourues est gardée en mémoire, de sorte que le réseau puisse s'entraîner puis valider son modèle. La répartition des solutions permettant l'apprentissage, puis la validation est de 60% - 40%, ce qui est tout a fait courant.

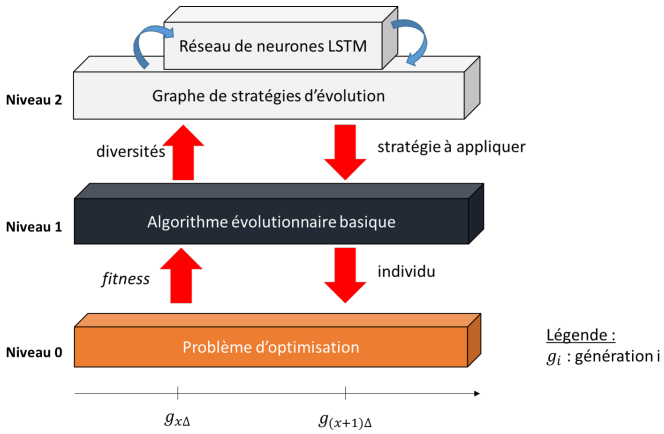


Figure 3.3 – Architecture de l'approche proposée. Il est constitué de 3 niveaux : le niveau 0 représente le problème à résoudre, le niveau 1, l'AE classique appliquant les opérateurs d'évolution sur la population, et le niveau 2 permet de modifier la stratégie S_c appliquée par la couche 1 sur la population ainsi que de simuler les populations des stratégies non sélectionnées sur Δ générations.

3.3.1 Initialisation

L'initialisation de la population est effectuée aléatoirement (selon une distribution uniforme) dans l'espace de recherche. Puis, le graphe est construit avec N noeuds, puis les poids sur les arcs sont initialisés de manière équiprobable, tel que $\omega_{i-j} = \frac{1}{N}$, avec $i - j$ l'arc orienté du noeud S_i vers le noeud S_j. Enfin, la première stratégie à appliquée est choisie aléatoirement. De plus, le modèle est créé à partir des premiers individus issus de la population initiale.

3.3.2 Phase d'évolution

Après l'initialisation, l'algorithme passe directement à la phase d'évolution. Celle-ci va entrer dans un cycle d'évolution et donc va répéter les mêmes étapes jusqu'à atteindre le ou les conditions d'arrêt. Ces étapes sont, dans l'ordre, l'application de la stratégie choisie S_c pendant Δ générations, la mémorisation des nouveaux individus générés au fur et à mesure, le modèle issu du réseau de neurones est mis à jour, les stratégies non appliquées pendant l'intervalle de générations précédent sont ensuite simulées, les poids $\omega_{S_iS_j}$ sont mis à jour, et enfin la nouvelle stratégie est sélectionnée. Il est important de rappeler qu'une seule population est considérée lors

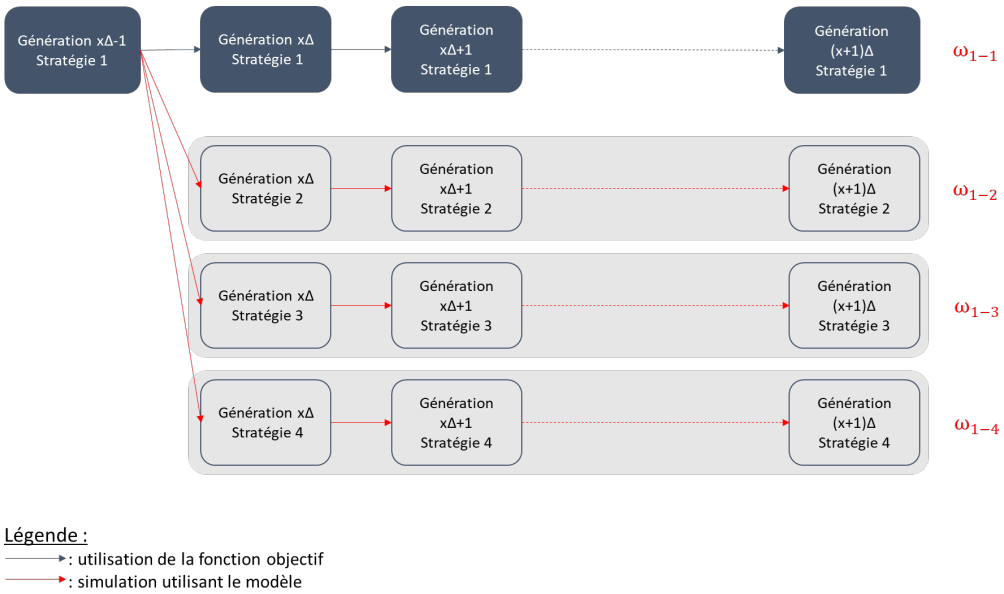


Figure 3.4 – Exemple avec $N = 4$ stratégies de la phase d'évolution. La stratégie appliquée pendant Δ générations à la population est $S_c = 1$. Puis, après mise à jour du modèle, les autres stratégies sont simulées parallèlement aux générations issues de S_c, les diversités sont ainsi mesurées pour mettre à jour les poids ω_{i-j}.

de la convergence, ce sont les diversités qui sont utiles à l'orientation de la recherche. Les étapes sont illustrées par un exemple avec $N = 4$ stratégies, voir la figure 3.4, et les détails par la suite.

Ainsi, nous reprenons l'exemple du cas où $N = 4$ stratégies. EAOG applique la stratégie sélectionnée S_c pendant les générations Δ sur la population. Puis, l'ensemble des nouveaux individus générés sur cet intervalle sont mémorisés, et le réseau de neurones LSTM peut être réentrainé, et le modèle amélioré. Ensuite, comme le montre la figure 3.4, les stratégies n'ayant pas été sélectionnées pour l'intervalle précédent sont simulées parallèlement grâce au modèle. Ainsi les diversités de chaque génération de chaque stratégie peut être mesurée. Si l'on suppose que la stratégie appliquée est $S_c = 1$, alors il est possible de mettre à jour tous les poids ω_{1-j} du graphe. Il s'agit alors de choisir le prochain noeud, pour cela le processus de sélection de la prochaine stratégie S_c est enclenchée. En effet, une composante stochastique est introduite dans la sélection, afin d'éviter de favoriser toujours les mêmes stratégies, tout en respectant les proportions de chaque ω_{1-j}. Ainsi, un nombre aléatoire r est tiré entre 0 et 1, une somme cumulée est des ω_{1-j} effectuée, et lorsque la somme est

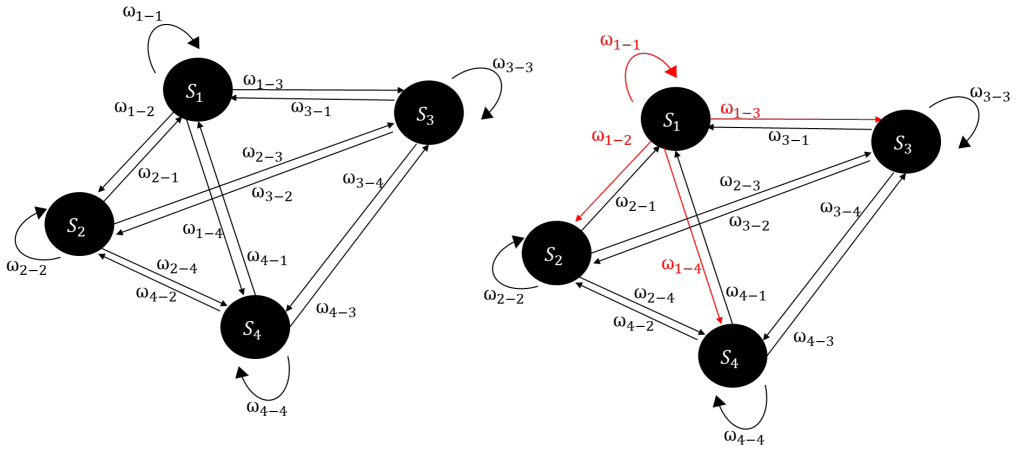


Figure 3.5 – Exemple avec $N = 4$ stratégies d'un graphe de stratégies d'évolution. Chaque noeud correspond à une stratégie, ici il y a donc les noeuds notés S_1, S_2, S_3et S_4. Cette figure met en évidence, en rouge sur le graphe de droite, les poids modifiés après la mise à jour du modèle dans le cas où $S_c = 1$.

Notation	Définition
D	Taille de l'individu ou dimension
P	Taille de la population
P_c	Taux de croisement
P_m	Taux de mutation
Δ	Intervalle de générations entre chaque changement de stratégie
N	Nombre de stratégies
Ω	Ensemble des paramètres relatifs au réseau de neurones LSTM
FEs	Nombre maximum d'évaluations

Tableau 3.1 – Sommaire des paramètres du EAOG.

supérieure à r, alors la stratégie S_c est celle correspondant à la stratégie du dernier poids ajouté dans la somme.

3.4 Analyse des sensibilités

Tests en cours

L'ensemble des paramètres du EAOG est résumé dans le tableau 3.1. De plus, les méthodes de croisement et de mutation composant l'ensemble des N stratégies mis à disposition de l'algorithme, sont les 20 même que celles utilisées pour le MEA.

Les méthodes de croisement sont donc : le croisement BLX-α, le croisement discret, le croisement linéaire, le croisement *one-point*, et le croisement barycentrique. Les méthodes de mutation sélectionnées sont : la mutation de Levy, la mutation Gaussienne, la mutation *scramble* et la mutation DE/RAND/1/BIN.

3.5 Conclusion

Une seconde métaheuristique adaptative basée sur la population a été proposée et détaillée dans ce chapitre. Ce nouvel algorithme, appelé EAOG, est capable de s'adapter tout au long du processus en choisissant la meilleure stratégie à appliquer. La stratégie choisie en parcourant un graphe basé sur les probabilités de diversité passées pour chaque stratégie. En plus des paramètres courants des algorithmes évolutionnaires, il y a deux paramères spécifiques au EAOG : l'intervalle entre les changements de stratégie tous les Δ générations, et le nombre de stratégies N disponibles.
Les expérimentations effectuées mettent en avant les différentes sensibilités des paramètres.
Le chapitre suivant présente l'analyse des performances des deux approches présentées dans cette thèse. Trois expériences ont permis de tester leurs performances : un ensemble de fonctions d'optimisation continue, sur un problèmes réel, et une étude de leur complexité.

Chapitre 4

Analyse des performances

4.1 Introduction

Dans ce chapitre, nous présentons et analysons les performances des algorithmes MEA et EAOG présentés dans les deux chapitres précédents. Tout d'abord, après avoir préciser les fonctions d'optimisation et les paramétrages, nous analysons statistiquement les performances des approches proposées comparées à celles de 13 autres algorithmes sur un ensemble à 34 fonctions d'optimisation. Ensuite, nous comparons les résultats obtenus sur le problème réel du cluster atomique de Lennard-Jones. Pour finir, nous comparons l'évolution des complexités des temps d'exécution selon la méthode proposée par le CEC 2015 [70].

4.2 Benchmark d'optimisation continue

4.2.1 Fonctions et paramétrage des algorithmes

Afin d'évaluer les algorithmes, 34 fonctions sont utilisées, notées de F_1 à F_{34}, en 2, 30 et 40 dimensions, voir [123] pour F_1 à F_{14} et F_{34}, [110] pour F_{15} à F_{17}, [63] pour F_{18} à F_{22} et F_{33}, [3] pour F_{23} à F_{32}. Ces fonctions sont présentées dans les annexes, les tableaux 4.22 à 4.25 détaillent le nom, l'espace de recherche, et l'optimum global à atteindre (si connu). Ces fonctions sont connues de la littérature et présentent des propriétés différentes afin de diversifier la difficulté de résolution, et de mettre en avant les faiblesses, ou au contraire l'efficacité et la stabilité de chacun des 15 algorithmes testés.
La comparaison des perfomances est basée sur 13 autres algorithmes issus de la littérature, qui sont : HdEA [18], RCGA-UNDX [84], CMA-ES [51], DE [106, 108], ODE [90], DEahcSPX [83, 116], DPSO [122], SEPSO [65], EDA [95], CoDE [117], JADE [128], jDE [15] et MPEDE [120]. Ces algorithmes sont choisis car ils représentent

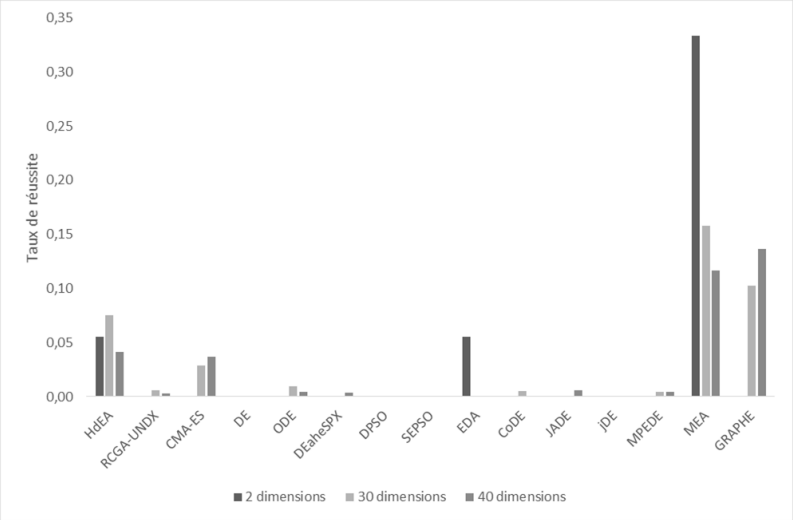


Figure 4.1 – Comparaison des taux de réussite des 15 algorithmes testés sur le benchmark de 34 problèmes d'optimisation en 2, 30 et 40 dimensions.

un ensemble de métaheuristiques variées, certaines plutôt classiques, ou anciennes, d'autres plus complexes, ou modernes. Les paramétrages des 15 algorithmes sont détaillés dans le tableau 4.1, et sont tirés de la littérature.
Le critère d'arrêt est le nombre maximum d'évaluations (FEs) fixé à 40000 en 30 et 40 dimensions pour les fonctions F_1 à F_{10} et F_{15} à F_{34}, et 1000 évaluations maximum en 2 dimensions pour les fonctions F_{11} à F_{14}. Pour effectuer une comparaison équitable, FEs est le même pour tous les algorithmes. Les expériences sont effectuées sur 100 lancements, et les résultats sont détaillés dans les annexes du tableau 4.26 au tableau 4.33. Pour MEA et EAOG, chaque lancement commence par une initialisation aléatoire de la population.

4.2.2 Comparaison des performances

Afin de comparer les performances des algorithmes nous avons étudié deux aspects : le taux de réussite, les distributions des rangs, et les statistiques tirées du test de Bonneferoni-Dunn.
La distribution des taux de réussite pour chaque dimension des 15 algorithmes est décrite par la figure 4.1. Le taux de réussite, noté P_s, est défini par le nombre d'occurences au rang 1 de l'algorithme, sur la somme des rangs qu'il a atteint. Par exemple, en 2 dimensions sur les 4 fonctions testées, MEA obtient 2 fois le rang 1 et 2 fois le

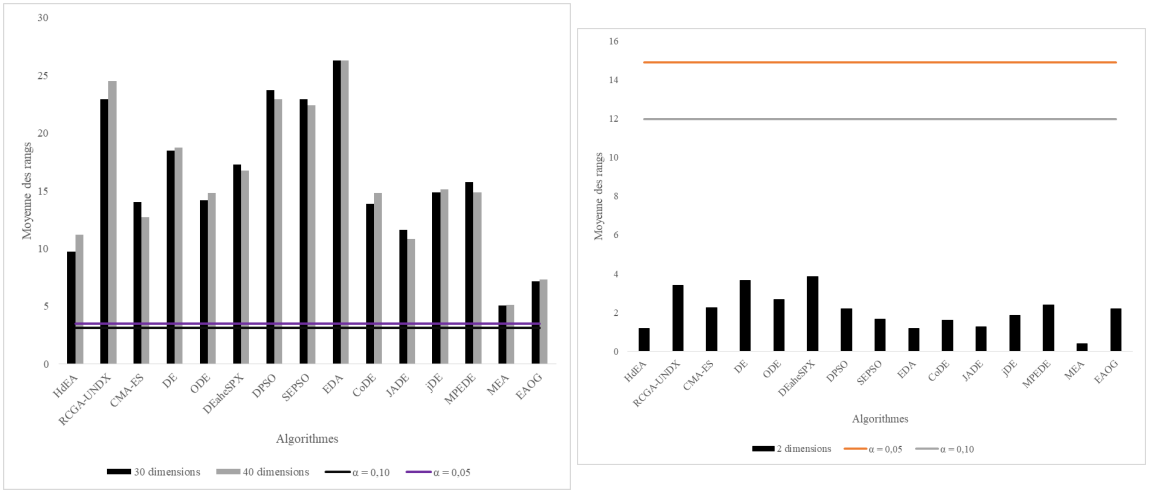


Figure 4.2 – Comparaison des différences critiques et classement des performances moyennes pour les 15 algorithmes en 30 et 40 dimensions (à gauche) et en 2 dimensions (à droite).

rang 2, le résultat obtenu est le suivant :
$P_s(MEA) = \frac{2}{6} = 0.33$
En 2 dimensions, 3 algorithmes obtiennent un taux de réussite non nul : HdEA (0.06%), EDA (0.06%) et MEA (0.33%). De même, en 30 dimensions, nous observons : RCGA-UNDX (0.01%), ODE (0.01%), CMA-ES (0.03%), HdEA (0.08%), EAOG (0.10%) et MEA (0.16%). MEA et EAOG ont les deux meilleurs taux de réussite, et ils sont supérieurs aux 4 autres algorithmes ayant un taux de réussite non nul. Enfin, en 40 dimensions cette fois nous relevons : JADE (0.01%), CMA-ES (0.04%), HdEA (0.04%), MEA (0.12%), et EAOG (0.14%). Là encore, EAOG et MEA sont les deux mieux classés, avec une nette différence de 10% comparée aux performances des autres algorithmes.
En observant ces résultats, nous pouvons affirmer que les deux approches proposées ont les meilleurs taux de réussite, avec des écarts parfois significatifs avec les autres algorithmes. De plus, la position des MEA et EAOG s'échangent entre première et deuxième position avec l'augmentation de la dimension.

De plus, lorsque l'on s'intéresse aux distributions respectives des rangs de MEA et de EAOG, deux comportements différents sont observés. En effet, là où MEA a une distribution relativement solide autour des rangs 1, 2, voir 3, et le plus haut rang atteint à 8, nous observons une distribution plus hétérogène pour EAOG qui

elle balaye tous les rangs, même si les occurences sont plus importantes pour les premiers rangs.
Pour aller plus en détails, en 2 dimensions MEA atteint la 50% du temps le rang 1 et 50% du temps le rang 2, là où EAOG atteint les rangs 6, 8, 9 et 10. Puis en 30 dimensions comme en 40 dimensions, les deux approches peut observer que les rangs 1 et 2 occupent la moitié de la distribution.

On peut en déduire que MEA et EAOG sont des algorithmes efficaces. Tout d'abord, parce que leur taux de réussite est supérieur à tous les autres algorithmes (supérieur de 10% en moyenne), et enfin parce qu'en 30 et 40 dimensions les deux premiers rangs sont atteints plus de la moitié du temps, pour MEA cela représente de 54% à 59% du temps, et pour EAOG 55% à 64%. De plus, en 2 dimensions, montre une performance particulière importante avec 33% de taux de réussite.

Une autre manière de mesurer l'efficacité des métaheuristiques étant d'employer une approche statistique, grâce au test de Bonferroni-Dunn. La première étape consiste à calculer la différence critique CD_α :

$$CD_\alpha = Q_\alpha \times \sqrt{\frac{g(g+1)}{6N}} \tag{4.1}$$

avec Q le coefficient, g le nombre de groupes (ou d'algorithmes), et N le degré de liberté (ou le nombre de fonctions).
En 30 et 40 dimensions, $g = 15$ et N=30 :

— pour $\alpha = 0.05$, $Q_\alpha = 2.997$ donc $CD_{\alpha=0.05} = 3.460$
— pour $\alpha = 0.10$, $Q_\alpha = 2.680$ donc $CD_{\alpha=0.10} = 3.094$

En 2 dimensions, avec $N = 4$:

— pour $\alpha = 0.05$, $Q_\alpha = 4.707$ donc $CD_{\alpha=0.05} = 14.884$
— pour $\alpha = 0.10$, $Q_\alpha = 3.787$ donc $CD_{\alpha=0.10} = 11.975$

Les rangs obtenus en premier lieu permettent de calculer la moyenne de classement (en divisant la somme de tous les rangs obtenus pour chaque algorithme par le nombre total de fonctions). En 2 dimensions, il n'y a pas de différence critique entre les résultats des algorithmes, leurs performances sont équivalentes, voir la figure 4.2 (à droite). Néanmoins, en dimensions 30 et 40, MEA est le meilleur que tous les autres algorithmes, suivi de près par EAOG, voir la figure 4.2 (à gauche). Ce qui

confirme les observations faîtes précédemment concernant les taux de réussite. Et qu'il s'agit bien de deux métaheuristiques robustes et efficaces sur une grande variété de problèmes continus.

4.3 Problème réel : cluster atomique de Lennard-Jones

Le problème du cluster atomique de Lennard-Jones (LJ) considère une molécule de K atomes en 3 dimensions dans l'espace de recherche avec le potentiel d'énergie minimal LJ. Les chercheurs de différents domaines scientifiques sont intéressés par la résolution de ce problème car il s'agit d'une fonction non convexe avec plusieurs optima locaux, et dont la complexité augmente de façon exponentielle $O(e^{N^2})$ avec N la dimension.
Mathématiquement le problème s'écrit comme suit :

$$MinV_k(x) = \sum_{i=1}^{K-1} \sum_{j=i+1}^{K} \left(\frac{1}{||x_i - x_j||^{12}} - \frac{1}{||x_i - x_j||^6}\right) \tag{4.2}$$

avec $i = 1, 2, ..., K$. La taille de la population est de 4, la dimension vaut 3×nombre d'atomes, $\Delta = 20$ generations, le domaine de définition est $[-2\,;\,2]$, et les résultats sont analysés sur 100 lancements. Le critère d'arrêt est de $65,000$ évaluations de la fonciton objectif. Les tests sot effectués pour 8, 9 et10 atomes, et les optima globaux sont respectivemenet de -19.821489, -24.113360 et -28.422532.
Les résultats du MEA et du EAOG sont comparés à ceux du SPSO2007 et de PSO-2S from [39]. Les SPSO2007 est en fait l'algorithme du Standard Particle Swarm Optimization (PSO) dans sa version de 2007, et le PSO-2S [40] est une variante du PSO qui s'appuie sur deux types d'essaims.
Pour 8 atomes, le SPSO2007 atteint 1.17 (écart-type de 1.8), le PSO-2S 1.47 ($9.12E-02$), le MEA $-2.31(4.26E-01)$ et le EAOG 0.00244219 ($9.05E-03$). Pour 9 atomes, le SPSO2007 obtient 2.57(2.99), le PSO-2S $2.31(6.06E-02)$, le MEA -2.55 ($4.06E-01$), et le EAOG 0.00004 ($1.76E-03$). Pour 10 atomes, le SPSO2007 a 4.66(4.22), le PSO-2S $1.52(2.28E-01)$, le MEA $-2.76(4.85E-01)$ et le EAOG 0.000065 ($1.01E-04$). Ainsi, les deux algorithmes proposés le MEA et le EAOG

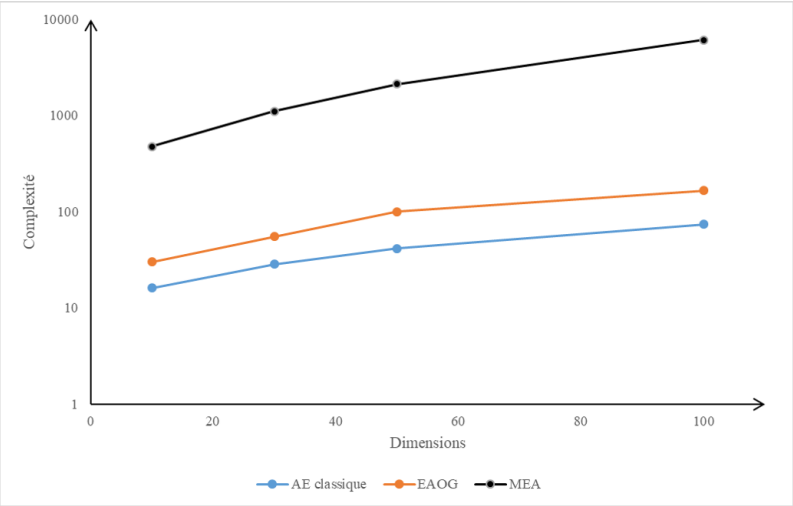


Figure 4.3 – Complexité des algorithmes EA classique, MEA et EAOG à l'échelle logarihmique.

obtiennent les meilleurs résultats pour 8, 9 et 10 atomes, ce qui met en avant leur efficacité.

4.4 Analyse de la complexité

Nous avons étudié la complexité du MEA et du EAOG en les comparant à celle d'un algorithme évolutionnaire classique en suivant la méthode du benchmark CEC 2015 [70]. La méthode de calcul se déroule suivant les étapes détaillées comme suit :

1. Calcul de T_0 : il s'agit de mesure le temps d'exécution nécessaire afin d'effectuer la boucle suivante :
   $for\ i = 1000000$:
   $x = 0.5 + (double)i$
   $x = x + x; x = x/2; x = x * x;$
   $x = sqrt(x); x = log(x);$
   $x = exp(x); x = x/(x + 2);$
   end
2. Calcul de T_1 : il s'agit d'exécuter l'algorithme sur la fonction $F1$ avec pour critère d'arrêt 200000 évaluations et de relever la durée d'exécution pour en déduire T_1 .

3. Calcul de $T2_{moy}$: la fonction $F1$ est lancée 5 fois avec toujours pour critère d'arrêt 200000 évaluations, puis la durée moyenne d'exécution $T2_{moy}$ est déduite.
4. Calculer du rapport $\frac{T2_{moy}-T_1}{T_0}$.

Cette procédure est répétée en 10, 30, 50 et 100 dimensions. Les résultats obtenus sont indiqués dans les tableaux 4.34 4.35 et 4.36, de plus la figure 4.3 illustre les différentes évolutions obtenues des temps d'exécution de chaque algorithme.
La figure 4.3 montre 3 courbes : la première courbe de couleur noire correspond aux résultats obtenus pour MEA, la seconde en orange est obtenue pour EAOG, et la dernière en bleu est obtenue pour un algorithme évolutionnaire classique. D'abord, toutes les complexités augmentent avec l'augmentation de la dimension. Cependant, les écart entre les évolutions observées sont relativement stables, c'est-à-dire que les distances entre les courbes sont conservées au fur et à mesure que les dimensions augmentent, même s'il semble que la distance en 100 dimensions soit légèrement plus importante concernant MEA par rapport aux deux autres approches. La complexité de MEA est d'ailleurs nettement supérieure à celles de EAOG, puis de l'AE classique. L'écart entre la comlpexité de EAOG et de l'AE classique est moins important et reste constant en fonction de la dimension.
Concernant MEA et EAOG, il est cohérent d'observer une augmentation de la complexité puisque nous avons ajouté dans l'architecture une couche de gestion, et même si cette augmentation reste stable en fonction de la dimension, l'impact sur les complexités initiales est bien plus significatif pour MEA que pour EAOG.

4.5 Conclusion

Dans ce dernier chapitre, nous avons analysé les performances des deux approches proposées dans cette thèse. Pour cela, nous avons testés MEA et EAOG sur un ensemble de fonctions d'optimisation continue, puis le problème du cluster atomique de Lennard-Jones, et enfin nous avons effectué un test pour en déduire la complexité des deux algorithmes en la comparant à celle d'un algorithme évolutionnaire classique.
La première expérience a permis à la fois de tester les deux approches, mais aussi de les comparer à 13 autres algorithmes de la littérature. Nous avons pu mettre en évidence l'efficacité des deux algorithmes d'une part, ainsi que les différences de

comportement d'autre part. En effet, l'analyse des taux de réussite, des distributions des rangs, ainsi que du test de Bonneferoni-Dunn, montrent les bons résultats obtenus, ainsi que l'amélioration des performances de EAOG au fur et à mesure que la dimension augmente, et inversement pour MEA. Ce qui a ensuite été confirmé par le test sur le problème réel du cluster atomique de Lennard-Jones.
Enfin, l'analyse des complexités de MEA, EAOG et EA classique, a permis de mettre en avant l'impact de l'ajout d'une couche de gestion dans l'architecture, cependant la complexité de MEA est nettement supérieure à celle de EAOG et de EA classique, même si elles restent toutes les trois relativement parallèle avec la dimension qui augmente.

Algorithme	Réglage des paramètres
HdEA	Le taux de croisement est égal à 0.1 et la taille de la population à 20 individus.
RCGA-UNDX	Le taux de croisement est égal à 0.1, l'opérateur UNDX est en mode trois parents. Pour les onfctions unimodales, la taille de la population est 50, sinon elle est de 300.
CMA-ES	La taille de la population vaut $4 + (3lnD)$, avec D la dimension.
DE	Le taux de croisement est 0.95, le facteur amplification vaut 0.5, la taille de population est 100, et l'opérateur de mutation est le DE/RAND/1/BIN.
ODE	The jumping rate is set to 0.3, les autres paramètres sont les même que pour le DE.
DEaheSPX	Le taux de croisement est 0.9, le facteur d'amplification est 0.9, la population est égale à la dimension, et le nombre de parents dans la métohde SPX est porté à 3.
DPSO	c_1 et c_2 sont égaux à 2, $C_v = 0.001$ et $C_m = 0.002$. ω est linéairement décroissant de 0.9 à 0.4, la taille de l'essaim est 20, la vélocité maximum V_{max} est $0.1R$ où $R = max(U_i - L_i)$ avec U_i la borne supérieure de la $i$ème dimension et L_i la borne inférieure respectivement.
SEPSO	c_1 et c_2 sont égaux à 2. ω est linéaire décroissant de 0.9 à 0.4, la taille de l'essaim s'élève à 20, la vélocité maximum $V_{max} = 0.1R$ avec $R = max(U_i - L_i)$, U_i la borne supérieure la $i$ème dimension et L_i la borne inférieure respectivement. Le facteur de rebondissement vaut -1.
EDA	La sélection élitiste $(\eta+\eta)$ est utilisée.
CoDE	La population est composée de 30 individus.
JADE	La taille de la population est de 100 individus.
jDE	Il y a 100 individus, le facteur d'échelle F vaut 0.5, le taux de croisement P_c égal 0.9.
MPEDE	La population est composée de 250 individuas, $\lambda_1 = \lambda_2 = \lambda_3 = 0.2$, ng=20, la probabilité de croisment $P_c = 0.5$ et le facteur d'échelle F=0.5.
MEA	La taille de la population $P = 50$, le taux de croisement $P_c = 0.7$, le taux de mutation $P_m = 0.3$(valeur courante dans la littérature), Δ vaut 20 générations, et le nombre de stratégies N égal 20.
EAOG	La taille de la population $P = 50$, le taux de croisement $P_c = 0.6$, le taux de mutation $P_m = 0.4$, Δ vaut 20 générations, et le nombre de stratégies N égal 20, le LSTM est paramétré avec 50000 epochs, 3 couches 32-32-1, l'algorithme d'optimisation est ADAM avec un pas de $1E{-}2$.

Tableau 4.1 – Sommaire des réglages des algorithmes.

Conclusion générale et perspectives

Dans cette thèse, nous avons présenté deux nouveaux algorithmes évolutionnaires adaptatifs d'optimisation continue : MEA et EAOG. Nous avons détaillé leurs fonctionnements, étudié la sensibilité de leurs paramètres, ainsi que leurs performances sur plusieurs fonctions d'optimisation et leur complexité. Le but en concevant ces approches était de faciliter l'utilisation d'algorithmes évolutionnaires en appliquant les meilleurs opérateurs pour tout au long du processus de résolution d'un problème d'optimisation. Pour cela, les deux algorithmes proposés utilisent des mécanismes prédictifs, basés sur des méthodes d'apprentissage, et elles se servent de la diversité comme mesure de performance. Ainsi, les choix des opérateurs à appliquer sur la population est dirigé par la conservation de la diversité au cours de la recherche. De plus, une terminologie particulière a été définie, car les opérateurs évolutionnaires sont combinés afin de créer un ensemble, de ce que nous avons appelé, des *stratégies* d'évolution. Dans ce travail, une stratégie d'évolution correspondra donc à l'association d'une méthode de croisement avec une méthode de mutation.
La première approche, MEA, propose ainsi d'associer les stratégies d'évolution au principe du maximum a posteriori. Quant à la seconde approche, EAOG, celle-ci représente les passages possibles d'une stratégie à une autre via un graphe, et elle utilise un réseau de neurones LSTM pour calculer les poids sur les arcs de ce graphe.

MEA est un algorithme évolutionnaire qui se sert du principe du maximum a posteriori associé aux stratégies évolutionnaires afin d'explorer de l'espace des solutions. En effet, à intervalle régulier, l'algorithme va remettre en question la stratégie appliquée sur sa population, et pour cela il va réévaluer le potentiel de conservation de diversité de chaque stratégie. Pour cela, sur un intervalle de générations passées, noté Δ, chaque stratégie est simulée à partir des générations correspondant à la population de solutions courantes. A partir de ces populations, la diversité est mesurée, les probabilités issues des diversités sont déduites, et le maximum a posteriori appliqué. Les stratégies peuvent ainsi, à partir des Δ simulations de générations passées,

être classées en fonction de la probabilité qu'elles ont à maximiser la diversité sur un futur intervalle de Δ générations. Cependant, la stratégie ayant la probabilité la plus haute n'est pas systématiquement sélectionnée, afin un phénomène de répétitions, un mécanisme de sélection introduisant une part stochastique tout en conservant la distribution des probabilités est mis en place.
Il s'agit donc là d'un algorithme évolutionnaire basé sur un mécanisme prédictif rendu possible grâce à l'emploi d'une méthode d'apprentissage dans le processus de convergence.

Si EAOG a le même objectif et certains principes en commun avec MEA, il n'en reste pas moins que sa mise en oeuvre est différente. En effet, EAOG est composé d'un graphe de stratégies d'évolution, parcouru tout au long de la convergence. Le graphe possède des noeuds et des poids sur les arcs, représentant respectivement les stratégies et les probabilités de passer d'une stratégie à une autre. Là aussi, l'algorithme va remettre en jeu à intervalle régulier le choix de la stratégie appliquée sur la population, et pour sélectionner la prochaine stratégie à appliquer, il va devoir parcourir le graphe. L'enjeu va alors être de donner les "bonnes" valeurs aux poids sur les arcs, afin de parcourir au mieux le graphe pour converger.
Pour cela, les populations des stratégies sont simulées à partir d'un modèle de la fonction objectif établi par un réseau de neurones LSTM. Ces simulations sur un intervalle de générations, permettent d'établir les probabilités issues des diversités mesurées, et donc de mettre à jour les poids sur les arcs. Une fois cette mise à jour effectuée, le même processus de sélection établi précédemment est mis en oeuvre.
Là aussi, une méthode d'apprentissage introduite dans le processus de recherche de la solution, a permis à un algorithme évolutionnaire d'être rendu adaptatif.

Lors de l'étude des sensibilités, deux types de paramètres ont été étudié : ceux présents pour tout algorithme évolutionnaire (taille de la population, taux de croisement, taux de mutation), et ceux spécifiques aux approches proposées (le nombre de stratégies N et le nombre de générations par intervalle Δ).
Dans le cas de MEA, Δ est insensible aux autres paramètres, et l'algorithme obtient les meilleurs résultats lorsque celui-ci est égal à 20. Cependant, le nombre de stratégies N lui est très sensible, et il a été observé qu'un nombre $N_{optimal}$ pourrait être établi. Quant aux sensibilités des autres paramètres, plus la taille de la population

augmente et plus
Les taux de croisement et de mutation ... A finir quand résultats EAOG obtenus
Paragraphe sur les tests de sensibilites de EAOG

Les deux algorithmes ont été lancés sur un ensemble de fonctions d'optimisation permettant ainsi de tester leurs performances sur différents profils de fonctions objectif, et en différentes dimensions (2, 30 et 40). MEA et EAOG obtiennent les meilleurs résultats sur l'ensemble de fonctions d'optimisation continue présenté. L'étude statistique de Dunn-Bonneferoni, ainsi que l'analyse des taux de réussite, détaillent la répartition des performances : MEA est premier en 2, de même en 30 dimensions, suivi de EAOG, cependant en 40 dimensions EAOG devance MEA qui arrive second. Appliqués au problème réel du cluster atomique de Lennard-Jones, MEA et EAOG arrivent là aussi premier et second. L'étude de la complexité montre néanmoins que si MEA obtient de meilleurs résultats lors des tests effectués, sa complexité est nettement supérieure à celle de EAOG et d'un EA classique.

Dans des travaux futurs, il serait intéressant d'appliquer les deux solutions proposées sur d'autres problèmes tels que les problèmes de traitement d'images, par exemple le recalage, les problèmes d'optimisation de méthodes d'apprentissage, ou bien des problèmes en plus grandes dimensions, afin de pouvoir observer ses performances dans d'autres domaines d'application.

De même, en perspective, il pourrait être intéressant t'étendre la notion de stratégies d'évolution non plus à un couple de méthodes de croisement et de mutation, mais d'y ajouter la méthode de sélection par exemple. Cela augmenterait le nombre de stratégies initiales, et donc la complexité combinatoire lors de la sélection d'une stratégie, mais l'observation de l'impact sur les performances pour apporter un indice sur l'intérêt ou non d'ajouter d'autres opérateurs d'évolution dans la notion de stratégie.

De plus, les algorithmes proposés dans cette thèse sont mono-objectifs, or en optimisation il existe de nombreux problèmes qui sont caractérisés de multi-objectifs, c'est-à-dire que plusieurs fonctions objectif sont à considérer en même temps. Il pourrait être intéressant d'étudier des versions multi-objectives du MEA et du EAOG.

Pour cela, il existe déjà des techniques se fondant notamment sur le front de Pareto pour pouvoir adapter les algorithmes du mono-objectif au multi-objectif.

De plus, concernant uniquement EAOG, il pourrait être profitable de sélectionner d'autres méthodes d'apprentissage, par exemple d'autres types de réseaux de neurones que le LSTM, afin d'étudier l'impact sur les performances que cela peut avoir, et de les comparer aux résultats déjà obtenus.

En outre, toujours à propos de EAOG, une modification de la méthode de mise à jour des poids sur les arcs du graphe pourrait être étudiée, afin potentiellement d'améliorer ses performances, ou à tout le moins d'étudier l'impact de sa modification sur les résultats obtenus et les comparer aux précédents.

Bibliographie

[1] Abbass, H. A., 2002. The self-adaptive pareto differential evolution algorithm. In : Proceedings of the IEEE Congress on Evolutionary Computation. pp. 831–836.

[2] Aickelin, U., Bentley, P., Cayzer, S., Kim, J., McLeod, J., 2003. Danger theory : The link between AIS and IDS ? In : Proceedings of the International Conference on Artificial Immune Systems. pp. 147–155.

[3] Ali, M. M., Khompatraporn, C., Zabinsky, Z. B., 2005. A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems. Journal of Global Optimization 31 (4), 635–672.

[4] Ali, M. Z., Awad, N. H., Suganthan, P. N., Reynolds, R. G., 2017. An adaptive multipopulation differential evolution with dynamic population reduction. IEEE Transactions on Cybernetics 47 (9), 2768–2779.

[5] Baluja, S., 1994. Population-based incremental learning. a method for integrating genetic search based function optimization and competitive learning. Tech. rep., Carnegie-Mellon University of Pittsburgh.

[6] Banks, A., Vincent, J., Anyakoha, C., 2007. A review of particle swarm optimization. Part I : background and development. Natural Computing 6 (4), 467–484.

[7] Battiti, R., 1996. Reactive search : toward self-tuning heuristics. Modern Heuristic Search Methods, 61–83.

[8] Bhanu, B., Lee, S., Ming, J., 1995. Adaptive image segmentation using a genetic algorithm. IEEE Transactions on Systems, Man, and Cybernetics 25 (12), 1543–1567.

[9] Bishop, C., Bishop, C. M., 1995. Neural networks for pattern recognition. Oxford University Press.

[10] Blayo, F., Verleysen, M., 1996. Les réseaux de neurones artificiels. Presses Universitaires de France.

[11] Blickle, T., Thiele, L., 1995. A comparison of selection schemes used in genetic algorithms. Tech. rep., Swiss Federal Institute of Technology.

[12] Bonabeau, E., Dorigo, M., Theraulaz, G., 1999. L'intelligence en essaim. In : Proceedings of the JFIADSMA. pp. 25–38.

[13] Boussaïd, I., Lepagnot, J., Siarry, P., 2013. A survey on optimization metaheuristics. Information Sciences 237, 82–117.

[14] Brest, J., Bošković, B., Greiner, S., Žumer, V., Maučec, M. S., 2007. Performance comparison of self-adaptive and adaptive differential evolution algorithms. Soft Computing - A Fusion of Foundations, Methodologies and Applications 11 (7), 617–629.

[15] Brest, J., Greiner, S., Boskovic, B., Mernik, M., Zumer, V., 2006. Self-adapting control parameters in differential evolution : A comparative study on numerical benchmark problems. IEEE Transactions on Evolutionary Computation 10 (6), 646–657.

[16] Caraffini, F., Neri, F., Picinali, L., 2014. An analysis on separability for memetic computing automatic design. Information Sciences 265, 1–22.

[17] Cayzer, S., Aickelin, U., 2002. Recommender system based on the immune network. In : Proceedings of the IEEE Conference on Evolutionary Computation.

[18] Chow, C. K., Yuen, S. Y., 2011. An evolutionary algorithm that makes decision based on the entire previous search history. IEEE Transactions on Evolutionary Computation 15 (6), 741–769.

[19] Clerc, M., 2003. TRIBES. Un exemple d'optimisation par essaim particulaire sans paramètres de contrôle. Tech. rep.

[20] Clerc, M., 2010. Particle swarm optimization. Vol. 93. John Wiley & Sons.

[21] Clerc, M., Kennedy, J., 2002. The particle swarm-explosion, stability, and convergence in a multidimensional complex space. IEEE Transactions on Evolutionary Computation 6 (1), 58–73.

[22] Clevert, D.-A., Unterthiner, T., Hochreiter, S., 2016. Fast and accurate deep network learning by exponential linear units (ELUs). In : Proceedings of the International Conference on Learning Representations. pp. 1–14.

[23] Cooren, Y., Clerc, M., Siarry, P., 2009. Performance evaluation of TRIBES, an adaptive particle swarm optimization algorithm. Swarm Intelligence 3 (2), 149–178.

[24] Cornuéjols, A., Miclet, L., Kodratoff, Y., 2002. Apprentissage artificiel : Concepts et algorithmes. Eyrolles.

[25] Dantzig, G. B., 1963. Linear programming and extensions. Princeton University Press Princeton.

[26] Darwin, C., 2004. On the origin of species, 1859. Routledge.

[27] Dasgupta, D., Forrest, S., 1999. Artificial immune systems in industrial applications. In : Proceedings of the second International Conference on Intelligent Processing and Manufacturing of Materials. Vol. 1. IEEE, pp. 257–267.

[28] Dasgupta, D., Yu, S., Nino, F., 2011. Recent advances in artificial immune systems : models and applications. Applied Soft Computing 11 (2), 1574–1587.

[29] De Castro, L. N., Von Zuben, F. J., 2002. Learning and optimization using the clonal selection principle. IEEE Transactions on Evolutionary Computation 6 (3), 239–251.

[30] Deneubourg, J.-L., Aron, S., Goss, S., Pasteels, J. M., 1990. The self-organizing exploratory pattern of the argentine ant. Journal of Insect Behavior 3 (2), 159–168.

[31] Di Caro, G., Dorigo, M., 1998. Antnet : distributed stigmergetic control for communications networks. Journal of Artificial Intelligence Research 9, 317–365.

[32] Dorigo, M., Gambardella, L. M., 1997. Ant colony system : a cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computation 1 (1), 53–66.

[33] Dréo, J., Pétrowski, A., Siarry, P., Taillard, E., 2003. Métaheuristiques pour l'optimisation difficile. Eyrolles.

[34] Dreyfus, G., Martinez, J., Samuelides, M., Gordon, M., Badran, F., Thiria, S., Hérault, L., 2002. Réseaux de Neurones, Méthodologie et Applications. Eyrolles.

[35] Eberhardt, B., Weber, A., Strasser, W., 1996. A fast, flexible, particle-system model for cloth draping. IEEE Computer Graphics and Applications 16 (5), 52–59.

[36] Eberhart, R., Kennedy, J., 1995. A new optimizer using particle swarm theory. In : Proceedings of the sixth International Symposium on Micro Machine and Human Science. pp. 39–43.

[37] Eglese, R., 1990. Simulated annealing : a tool for operational research. European Journal of Operational Research 46 (3), 271–281.

[38] Eiben, Á. E., Hinterding, R., Michalewicz, Z., 1999. Parameter control in evolutionary algorithms. IEEE Transactions on Evolutionary Computation 3 (2), 124–141.

[39] El Dor, A., Clerc, M., Siarry, P., 2012. A multi-swarm PSO using charged particles in a partitioned search space for continuous optimization. Computational Optimization and Applications 53 (1), 271–295.

[40] El Dor, A., Lepagnot, J., Nakib, A., Siarry, P., 2014. PSO-2S optimization algorithm for brain MRI segmentation. Springer, pp. 13–22.

[41] Eshelman, J. L., Schaffer, J. D., 1993. Real-coded genetic algorithms and interval-schema. Foundation of Genetic Algorithms 2, 187–202.

[42] Farmer, J. D., Packard, N. H., Perelson, A. S., 1986. The immune system, adaptation, and machine learning. Physica D : Nonlinear Phenomena 22 (1-3), 187–204.

[43] Feo, T. A., Resende, M. G., 1995. Greedy randomized adaptive search procedures. Journal of Global Optimization 6 (2), 109–133.

[44] Fogel, L. J., Owens, A. J., Walsh, M. J., 1966. Artificial intelligence through simulated evolution. John Wiley & Sons.

[45] Forrest, S., Perelson, A. S., Allen, L., Cherukuri, R., 1994. Self-nonself discrimination in a computer. In : Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy. IEEE, pp. 202–212.

[46] Förster, M., Bickel, B., Hardung, B., Kókai, G., 2007. Self-adaptive ant colony optimisation applied to function allocation in vehicle networks. In : Proceedings of the ninth annual Conference on Genetic and Evolutionary Computation. ACM, pp. 1991–1998.

[47] Glover, F., 1986. Future paths for integer programming and links to artificial intelligence. Computers and Operations Research 13, 533–549.

[48] Goldberg, D. E., Deb, K., Clark, J. H., 1991. Genetic algorithms, noise, and the sizing of populations. Urbana 51, 61801.

[49] Goldberg, D. E., Holland, J. H., 1988. Genetic algorithms and machine learning. Machine Learning 3 (2), 95–99.

[50] Greensmith, J., Aickelin, U., Cayzer, S., 2005. Introducing dendritic cells as a novel immune-inspired algorithm for anomaly detection. In : Proceedings of the International Conference on Artificial Immune Systems. Springer, pp. 153–167.

[51] Hansen, N., 2006. The CMA evolution strategy : a comparing review. Towards a new Evolutionary Computation, 75–102.

[52] Hansen, N., Müller, S. D., Koumoutsakos, P., 2003. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). IEEE Transactions on Evolutionary Computation 11 (1), 1–18.

[53] Harik, G. R., Lobo, F. G., Goldberg, D. E., 1999. The compact genetic algorithm. IEEE Transactions on Evolutionary Computation 3 (4), 287–297.

[54] He, K., Zhang, X., Ren, S., Sun, J., 2015. Delving deep into rectifiers : surpassing human-level performance on imagenet classification. In : Proceedings of the IEEE International Conference on Computer Vision. pp. 1026–1034.

[55] Heppner, F., Grenander, U., 1990. A stochastic nonlinear model for coordinated bird flocks. AAAS, pp. 233–238.

[56] Hillis, W. D., 1990. Co-evolving parasites improve simulated evolution as an optimization procedure. Physica D : Nonlinear Phenomena 42 (1-3), 228–234.

[57] Hinterding, R., November 1995. Gaussian mutation and self-adaption for numeric genetic algorithms. In : Proceedings of the IEEE International Conference on Evolutionary Computation. p. 384.

[58] Hochreiter, S., Schmidhuber, J., 1997. Long short-term memory. Neural Computation 9 (8), 1735–1780.

[59] Holland, J. H., 1992. Adaptation in natural and artificial systems : an introductory analysis with applications to biology, control, and artificial intelligence. MIT press.

[60] Iacca, G., Caraffini, F., Neri, F., 2014. Multi-strategy coevolving aging particle optimization. International Journal of Neural Systems 24 (1), 1450008 [19 pages].

[61] Jerne, N. K., 1973. The immune system. Scientific American 229 (1), 52–63.

[62] Kantorovich, L. V., 1960. Mathematical methods of organizing and planning production. Management Science 6 (4), 366–422.

[63] Koumousis, V. K., Katsaras, C. P., 2006. A sawtooth genetic algorithm combining the effects of variable population size and reinitialization to enhance performance. IEEE Transactions on Evolutionary Computation 10 (1), 19–28.

[64] Koza, J. R., 1992. Genetic programming II, automatic discovery of reusable subprograms. MIT Press.

[65] Krink, T., Vesterstrøm, J. S., Riget, J., August 2002. Particle swarm optimisation with spatial particle extension. In : Proceedings of the IEEE Congress on Evolutionary Computation. Vol. 2. pp. 1474–1479.

[66] Larranaga, P., 2002. A review on estimation of distribution algorithms. Springer, pp. 57–100.

[67] Larrañaga, P., Kuijpers, C. M. H., Murga, R. H., Inza, I., Dizdarevic, S., 1999. Genetic algorithms for the travelling salesman problem : a review of representations and operators. Artificial Intelligence Review 13 (2), 129–170.

[68] Lee, C.-Y., Yao, X., 2004. Evolutionary programming using mutations based on Levy probability distribution. IEEE Transactions on Evolutionary Computation 8 (1), 1–13.

[69] Lettvin, J. Y., Maturana, H. R., McCulloch, W. S., Pitts, W. H., 1959. What the frog's eye tells the frog's brain. In : Proceedings of the IRE. Vol. 47. IEEE, pp. 1940–1951.

[70] Liang, J., Qu, B., Suganthan, P., Chen, Q., 2014. Problem definitions and evaluation criteria for the CEC 2015 competition on learning-based real-parameter single objective optimization. Tech. rep., Nanyang Technological University.

[71] Liu, J., Lampinen, J., 2005. A fuzzy adaptive differential evolution algorithm. Soft Computing 9 (6), 448–462.

[72] Loshchilov, I., 2013. Cma-es with restarts for solving CEC 2013 benchmark problems. In : Proceedings of the IEEE Congress on Evolutionary Computation. IEEE, pp. 369–376.

[73] Lozada-Chang, L.-V., Santana, R., 2011. Univariate marginal distribution algorithm dynamics for a class of parametric functions with unitation constraints. Information Sciences 181 (11), 2340–2355.

[74] Mantegna, R. N., 1994. Fast, accurate algorithm for numerical simulation of Levy stable stochastic processes. Physical Review E 49 (5), 4677.

[75] McCulloch, W. S., Pitts, W., 1943. A logical calculus of the ideas immanent in nervous activity. Bulletin of Mathematical Biophysics 5 (4), 115–133.

[76] Mitchell, T. M., 1997. Machine learning. McGraw-Hill Education.

[77] Mohamed, A. W., Suganthan, P. N., 2017. Real-parameter unconstrained optimization based on enhanced fitness-adaptive differential evolution algorithm with novel mutation. Soft Computing, 1–21.

[78] Moscato, P., 1989. On genetic crossover operators for relative order preservation. Tech. Rep. 778, Caltech Concurrent Computation Program, Report C3P.

[79] Murata, Y., Shibata, N., Yasumoto, K., Ito, M., et al., 2002. Agent oriented self adaptive genetic algorithm. pp. 348–353.

[80] Neri, F., del Toro Garcia, X., Cascella, G. L., Salvatore, N., 2008. Surrogate assisted local search in pmsm drive design. COMPEL-The International Journal for Computation and Mathematics in Electrical and Electronic Engineering 27 (3), 573–592.

[81] Neri, F., Toivanen, J., Cascella, G. L., Ong, Y.-S., 2007. An adaptive multimeme algorithm for designing HIV multidrug therapies. IEEE/ACM Transactions on Computational Biology and Bioinformatics 4 (2).

[82] Nielsen, S. S., Torres, C. F., Danoy, G., Bouvry, P., 2016. Tackling the ifp problem with the preference-based genetic algorithm. In : Proceedings of the Genetic and Evolutionary Computation Conference. ACM, pp. 965–972.

[83] Noman, N., Iba, H., 2008. Accelerating differential evolution using an adaptive local search. IEEE Transactions on Evolutionary Computation 12 (1), 107–125.

[84] Ono, I., Kobayashi, S., July 1997. A real coded genetic algorithm for function optimization using unimodal normal distributed crossover. In : Proceedings of the seventh International Conference on Genetic Algorithms, San Francisco, USA. pp. 246–253.

[85] Oppacher, F., Wineberg, M., July 1999. The shifting balance genetic algorithm : Improving the GA in a dynamic environment. In : Proceedings of the first annual Conference on Genetic and Evolutionary Computation, Orlando, USA. Vol. 1. pp. 504–510.

[86] Poikolainen, I., Neri, F., Caraffini, F., 2015. Cluster-based population initialization for differential evolution frameworks. Information Sciences 297, 216–235.

[87] Poli, R., Langdon, W. B., 1998. Schema theory for genetic programming with one-point crossover and point mutation. Evolutionary Computation 6 (3), 231–252.

[88] Potter, M. A., De Jong, K. A., 1994. A cooperative coevolutionary approach to function optimization. In : Proceedings of the International Conference on Parallel Problem Solving from Nature. Springer, pp. 249–257.

[89] Price, K. V., Storn, R. M., Lampinen, J. A., 2005. The differential evolution algorithm. Springer, pp. 37–134.

[90] Rahnamayan, S., Tizhoosh, H. R., Salama, M. M., 2008. Opposition-based differential evolution. IEEE Transactions on Evolutionary computation 12 (1), 64–79.

[91] Rechenberg, I., 1973. Evolution strategy : optimization of technical systems by means of biological evolution. Fromman-Holzboog 104, 15–16.

[92] Rennard, J.-P., 2006. Réseaux neuronaux : une introduction accompagnée d'un modèle Java. Vuibert.

[93] Reynolds, C. W., 1987. Flocks, herds and schools : a distributed behavioral model. In : Proceedings of the 14th annual Conference on Computer Graphics and Interactive Techniques. Vol. 21. ACM, pp. 25–34.

[94] Rostami, S., Neri, F., 2016. Covariance matrix adaptation pareto archived evolution strategy with hypervolume-sorted adaptive grid algorithm. Integrated Computer-Aided Engineering 23 (4), 313–329.

[95] Santana, R., Bielza, C., Larranaga, P., Lozano, J. A., Echegoyen, C., Mendiburu, A., Armananzas, R., Shakya, S., 2010. Mateda-2.0 : Estimation of distribution algorithms in matlab. Journal of Statistical Software 35 (7), 1–30.

[96] Sawai, H., Adachi, S., 1999. Genetic algorithm inspired by gene duplication. In : Proceedings of the IEEE Congress on Evolutionary Computation. Vol. 1. IEEE, pp. 480–487.

[97] Schnecke, V., Vornberger, O., 1996. An adaptive parallel genetic algorithm for VLSI-layout optimization. In : Proceedings for the International Conference on Parallel Problem Solving from Nature. Springer, pp. 859–868.

[98] Schwefel, H.-P., 1981. Numerical optimization of computer models. John Wiley & Sons.

[99] Shi, Y., Eberhart, R. C., 2001. Fuzzy adaptive particle swarm optimization. In : Proceedings of the IEEE Congress on Evolutionary Computation. Vol. 1. IEEE, pp. 101–106.

[100] Shimodaira, H., November 1997. DCGA : A diversity control oriented genetic algorithm. In : Proceedings of the IEEE International Conference on Tools with Artificial Intelligence, Newport Beach, USA. pp. 367–374.

[101] Siarry, P., 2014. Métaheuristiques. Editions Eyrolles.

[102] Siarry, P., Berthiau, G., Durdin, F., Haussy, J., 1997. Enhanced simulated annealing for globally minimizing functions of many-continuous variables. ACM Transactions on Mathematical Software (TOMS) 23 (2), 209–228.

[103] Siegelmann, H. T., 1995. Computation beyond the turing limit. Science 268 (5210), 545–548.

[104] Simon, G., 2008. La planète migratoire dans la mondialisation. Armand Colin Paris.

[105] Stanley, K. O., Miikkulainen, R., 2004. Competitive coevolution through evolutionary complexification. Journal of Artificial Intelligence Research 21, 63–100.

[106] Storn, R., Price, K., 1995. Differential evolution : a simple and efficient adaptive scheme for global optimization over continuous spaces. Berkeley ICSI.

[107] Storn, R., Price, K., 1997. Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization 11 (4), 341–359.

[108] Storn, R., Price, K., 1997. Differential evolution : a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization 11 (4), 341–359.

[109] Suganthan, P. N., 1999. Particle swarm optimiser with neighbourhood operator. In : Proceedings of the IEEE Congress on Evolutionary Computation. Vol. 3. IEEE, pp. 1958–1962.

[110] Suganthan, P. N., Hansen, N., Liang, J. J., Deb, K., Chen, Y.-P., Auger, A., Tiwari, S., 2005. Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization. Tech. rep., Nanyang Technological University.

[111] Taillard, E. D., 1998. La programmation à memoire adaptative et les algorithmes pseudo-gloutons : nouvelles perspectives pour les meta-heuristiques. Tech. rep., Istituto Dalle Molle Di Studi Sull Intelligenza Artificiale.

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

[113] Timmis, J., Hone, A., Stibor, T., Clark, E., 2008. Theoretical advances in artificial immune systems. Theoretical Computer Science 403 (1), 11–32.

[114] Touzet, C., 1992. les réseaux de neurones artificiels, introduction au connexionnisme. EC2.

[115] Tsutsui, S., Fujimoto, Y., Ghosh, A., 1997. Forking genetic algorithms : GAs with search space division schemes. Evolutionary Computation 5 (1), 61–80.

[116] Tsutsui, S., Yamamura, M., Higuchi, T., July 1999. Multi-parent recombination with simplex crossover in real coded genetic algorithms. In : Proceedings of the first Annual Conference on Genetic and Evolutionary Computation, San Francisco, CA, USA. Vol. 1. Morgan Kaufmann Publishers Inc., pp. 657–664.

[117] Wang, Y., Cai, Z., Zhang, Q., 2011. Differential evolution with composite trial vector generation strategies and control parameters. IEEE Transactions on Evolutionary Computation 15 (1), 55–66.

[118] Wilson, E. O., MacArthur, R. H., 1967. The theory of island biogeography. Princeton University Press.

[119] Wright, A. H., 1991. Genetic algorithms for real parameter optimization. Foundations of Genetic Algorithms 1, 205–218.

[120] Wu, G., Mallipeddi, R., Suganthan, P. N., Wang, R., Chen, H., 2016. Differential evolution with multi-population based ensemble of mutation strategies. Information Sciences 329, 329–345.

[121] Xie, X.-F., Zhang, W.-J., Yang, Z.-L., 2002. Adaptive particle swarm optimization on individual level. In : Proceedings of the 6th International Conference on Signal Processing. Vol. 2. IEEE, pp. 1215–1218.

[122] Xie, X. F., Zhang, W. J., Yang, Z. L., May 2002. A discrete particle swarm optimization. In : Proceedings of the IEEE Congress on Evolutionary Computation, Hawaii, USA. IEEE, pp. 1666–1670.

[123] Yao, X., Liu, Y., Lin, G., 1999. Evolutionary programming made faster. IEEE Transactions on Evolutionary Computation 3 (2), 82–102.

[124] Yasuda, K., Iwasaki, N., 2004. Adaptive particle swarm optimization using velocity information of swarm. In : Proceedings of the IEEE International Conference on Systems, Man and Cybernetics. Vol. 4. IEEE, pp. 3475–3481.

[125] Yu, E., Suganthan, P. N., 2010. Ensemble of niching algorithms. Information Sciences 180 (15), 2815–2833.

[126] Zaharie, D., 2002. Critical values for the control parameters of differential evolution algorithms. In : Proceedings of the International Conference on Soft Computing (MENDEL). Vol. 2. pp. 62–67.

[127] Zaharie, D., 2003. Control of population diversity and adaptation in differential evolution algorithms. In : Proceedings of the International Conference on Soft Computing (MENDEL). Vol. 9. pp. 41–46.

[128] Zhang, J., Sanderson, A. C., 2009. JADE : adaptive differential evolution with optional external archive. IEEE Transactions on Evolutionary Computation 13 (5), 945–958.

[129] Zhang, W., Liu, Y., Clerc, M., 2003. An adaptive PSO algorithm for reactive power optimization. In : Proceedings of the 6th International Conference on Advances in Power System Control, Operation and Management. IET, pp. 302–307.

[130] Zhang, W.-J., Xie, X.-F., 2003. DEPSO : hybrid particle swarm with differential evolution operator. In : Proceedings of the IEEE International Conference on Systems, Man and Cybernetics. Vol. 4. IEEE, pp. 3816–3821.

Chapitre 4

Annexes

Sensibilités du MEA (tableaux 4.2 - 4.11)

Sensibilités du EAOG (tableaux 4.12 - 4.21)

Benchmark de 34 fonctions d'optimisation (tableaux 4.22 - 4.25)

Résultats expérimentaux obtenus pour le MEA et le EOAOG sur un benchmark de 34 fonctions (tableaux 4.26-4.33, les meilleurs résultats sont en gras)

Complexité concernant le MEA, le EAOG et un algorithme évolutionnaire classique (tableaux 4.34 - 4.36)

Dimension	20	50	100	200
$\Delta = 10$	$5.98E{-}01$ $(3.40E{+}00)$	$9.96E{-}01$ $(6.97E{+}00)$	$9.96E{-}01$ $(9.91E{+}00)$	$1.49E{-}03$ $(1.27E{-}02)$
$\Delta = 20$	**3.99E-01** **(2.79E+00)**	$3.63E{-}07$ $(3.61E{-}06)$	**2.83E-07** **(1.64E-06)**)	**1.33E-03** **(3.78E-03)**
$\Delta = 50$	$5.98E{-}01$ $(3.40E{+}00)$	**4.37E-09** **(4.00E-08)**	$1.01E{+}00$ $(9.90E{+}00)$	$1.04E{-}01$ $(8.62E{-}01)$
$\Delta = 100$	$8.99E{-}01$ $(3.98E{+}00)$	$4.89E{-}01$ $(4.95E{+}00)$	$2.34E{+}00$ $(1.40E{+}01)$	$3.78E{+}00$ $(2.22E{+}01)$
$\Delta = 200$	$1.00E{+}00$ $(4.34E{+}00)$	$2.51E{+}00$ $(1.08E{+}01)$	$3.36E{+}00$ $(1.58E{+}01)$	$1.01E{+}00$ $(5.33E{+}00)$
$\Delta = 500$	$8.90E{+}00$ $(1.15E{+}01)$	$8.90E{+}00$ $(2.61E{+}01)$	$3.31E{+}01$ $(1.22E{+}02)$	$9.99E{+}01$ $(3.41E{+}02)$

Tableau 4.2 – Résultats obtenus sur 100 lancements en fonction de Δ et de la dimension pour une population de 50 individus, sur la fonction Rastrigin et avec un nombre de générations maximum égale à 5000.

Dimension	20	50	100	200
$\Delta = 10$	$3.98E{-}01$ $(2.79E{+}00)$	**0.00E+00** **(0.00E+00)**	$9.96E{-}01$ $(9.91E{+}00)$	**1.35E-05** **(8.40E-05)**
$\Delta = 20$	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**	$1.28E{-}03$ $(1.23E{-}02)$
$\Delta = 50$	$3.98E{-}01$ $(2.79E{+}00)$	$9.96E{-}01$ $(6.97E{+}01)$	$1.01E{+}00$ $(9.91E{+}00)$	$3.00E{-}04$ $(2.90E{-}03)$
$\Delta = 100$	$1.99E{-}01$ $(1.98E{+}00)$	$5.89E{-}06$ $(3.37E{-}05)$	$4.85E{-}04$ $(2.51E{-}03)$	$3.17E{+}00$ $(2.25E{+}01)$
$\Delta = 200$	**0.00E+00** **(2.06E-08)**	$4.98E{-}01$ $(4.95E{+}00)$	$7.06E{-}03$ $(2.46E{-}02)$	$2.18E{+}00$ $(1.98E{+}01)$
$\Delta = 500$	$2.18E{+}00$ $(6.90E{+}00)$	$4.35E{+}00$ $(1.59E{+}01)$	$1.61E{+}01$ $(4.40E{+}01)$	$4.83E{+}01$ $(1.50E{+}02)$

Tableau 4.3 – Résultats obtenus sur 100 lancements en fonction de Δ et de la dimension pour une population de 100 individus, sur la fonction Rastrigin et avec un nombre de générations maximum égale à 5000.

Dimension	20	50	100	200
$\Delta = 10$	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**
$\Delta = 20$	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(1.92E-08)**
$\Delta = 50$	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**	$8.13E{-}06$ $(7.97E{-}05)$
$\Delta = 100$	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(1.12E-08)**	$1.70E{-}05$ $(1.10E{-}04)$
$\Delta = 200$	**0.00E+00** **(1.48E-08)**	**0.00E+00** **(0.00E+00)**	$3.72E{-}06$ $(3.70E{-}05)$	$4.68E{-}04$ $(4.11E{-}03)$
$\Delta = 500$	$1.31E{-}01$ $(8.10E{-}01)$	$3.44E{+}00$ $(1.12E{+}01)$	$1.56E{+}01$ $(4.32E{+}01)$	$8.41E{+}01$ $(1.92E{+}02)$

Tableau 4.4 – Résultats obtenus sur 100 lancements en fonction de Δ et de la dimension pour une population de 300 individus, sur la fonction Rastrigin et avec un nombre de générations maximum égale à 5000.

Dimension	20	50	100	200
$\Delta = 10$	**0.00E+00**	**0.00E+00**	**0.00E+00**	**1.37E-13**
	(0.00E+00)	**(0.00E+00)**	**(0.00E+00)**	**(7.78E-13)**
$\Delta = 20$	**4.38E-13**	**0.00E+00**	**0.00E+00**	**1.05E-15**
	(4.36E-12)	**(0.00E+00)**	**(0.00E+00)**	**(7.39E-15)**
$\Delta = 50$	**2.26E-13**	**3.84E-12**	**7.64E-14**	**2.88E-09**
	(2.24E-12)	**(3.82E-11)**	**(7.60E-13)**	**(2.86E-08)**
$\Delta = 100$	**1.44E-09**	$1.90E{-}07$	**0.00E+00**	**4.18E-10**
	(1.43E-08)	$(1.37E{-}06)$	**(0.00E+00)**	**(4.03E-09)**
$\Delta = 200$	$9.03E{-}02$	$7.84E{-}03$	$6.59E{-}01$	$6.65E{+}00$
	$(8.98E{-}01)$	$(7.77E{-}02)$	$(6.51E{+}00)$	$(6.48E{+}01)$
$\Delta = 500$	**1.11E-09**	$3.66E{-}01$	$5.91E{-}01$	$2.18E{+}01$
	(1.10E-08)	$(2.56E{+}00)$	$(5.73E{+}00)$	$(8.08E{+}01)$

Tableau 4.5 – Résultats obtenus sur 100 lancements en fonction de Δ et de la dimension pour une population de 500 individus, sur la fonction Rastrigin et avec un nombre de générations maximum égale à 5000.

Dimension	20	50	100	200
$\Delta = 10$	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	(0.00E+00)	**(0.00E+00)**	**(0.00E+00)**	**(0.00E+00)**
$\Delta = 20$	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	(0.00E+00)	**(0.00E+00)**	**(0.00E+00)**	**(0.00E+00)**
$\Delta = 50$	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	(0.00E+00)	**(0.00E+00)**	**(0.00E+00)**	**(0.00E+00)**
$\Delta = 100$	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	(0.00E+00)	**(0.00E+00)**	**(0.00E+00)**	**(0.00E+00)**
$\Delta = 200$	**0.00E+00**	**0.00E+00**	$4.49E{-}01$	$8.67E{-}05$
	(0.00E+00)	**(0.00E+00)**	$(4.47E{+}00)$	$(6.95E{-}04)$
$\Delta = 500$	$1.56E{-}06$	$2.77E{-}05$	$8.07E{-}01$	$1.11E{+}00$
	$(1.55E{-}05)$	$(2.73E{-}04)$	$(5.73E{+}00)$	$(1.06E{+}01)$

Tableau 4.6 – Résultats obtenus sur 100 lancements en fonction de Δ et de la dimension pour une population de 800 individus, sur la fonction Rastrigin et avec un nombre de générations maximum égale à 5000.

Dimension	20	50	100	200
$\Delta = 10$	$1.15E{-}03$	**0.00E+00**	**0.00E+00**	$1.19E{-}03$
	$(1.15E{-}02)$	**(0.00E+00)**	**(0.00E+00)**	$(1.57E{-}03)$
$\Delta = 20$	**0.00E+00**	$4.27E{-}04$	**0.00E+00**	$3.00E{-}04$
	(0.00E+00)	$(6.65E{-}04)$	**(0.00E+00)**	$(6.31E{-}04)$
$\Delta = 50$	$2.65E{-}03$	**0.00E+00**	**0.00E+00**	$1.52E{-}03$
	$(2.64E{-}02)$	**(0.00E+00)**	**(0.00E+00)**	$(3.30E{-}03)$
$\Delta = 100$	$1.70E{-}03$	**0.00E+00**	**0.00E+00**	$4.92E{-}04$
	$(1.69E{-}02)$	**(0.00E+00)**	**(0.00E+00)**	$(1.71E{-}03)$
$\Delta = 200$	$8.90E{-}03$	$2.80E{-}06$	$2.70E{-}05$	$2.89E{-}02)$
	$(4.00E{-}02)$	$(2.57E{-}05)$	$(1.97E{-}04)$	$(6.34E{-}02)$
$\Delta = 500$	$3.02E{-}03$	$7.13E{-}04$	$4.25E{-}02$	$3.32E{-}01$
	$(1.78E{-}02)$	$(2.84E{-}03)$	$(1.43E{-}01)$	$(3.06E{-}01)$

Tableau 4.7 – Résultats obtenus sur 100 lancements en fonction de Δ et de la dimension pour une population de 50 individus, sur la fonction Griewank et avec un nombre de générations maximum égale à 5000.

Dimension	20	50	100	200
$\Delta = 10$	**0.00E+00**	**0.00E+00**	**0.00E+00**	$1.67E{-}05$
	(0.00E+00)	**(0.00E+00)**	**(0.00E+00)**	$(2.75E{-}05)$
$\Delta = 20$	**0.00E+00**	**0.00E+00**	**0.00E+00**	$7.91E{-}05$
	(0.00E+00)	**(0.00E+00)**	**(0.00E+00)**	$(9.14E{-}05)$
$\Delta = 50$	$9.10E{-}04$	**0.00E+00**	**0.00E+00**	$2.45E{-}04$
	$(9.06E{-}03)$	**(0.00E+00)**	**(0.00E+00)**	$(4.26E{-}04)$
$\Delta = 100$	**0.00E+00**	**0.00E+00**	**0.00E+00**	$1.11E{-}03$
	(0.00E+00)	**(0.00E+00)**	**(0.00E+00)**	$(2.04E{-}03)$
$\Delta = 200$	**0.00E+00**	**0.00E+00**	$1.97E{-}08$	$1.12E{-}03$
	(0.00E+00)	**(0.00E+00)**	$(6.05E{-}08)$	$(5.97E{-}03)$
$\Delta = 500$	$2.95E{-}03$	$2.73E{-}04$	$2.36E{-}03$	$1.18E{-}02$
	$(1.51E{-}02)$	$(1.94E{-}03)$	$(1.69E{-}02)$	$(1.05E{-}01)$

Tableau 4.8 – Résultats obtenus sur 100 lancements en fonction de Δ et de la dimension pour une population de 100 individus, sur la fonction Griewank et avec un nombre de générations maximum égale à 5000.

Dimension	20	50	100	200
$\Delta = 10$	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	(0.00E+00)	**(0.00E+00)**	**(0.00E+00)**	**(0.00E+00)**
$\Delta = 20$	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	(0.00E+00)	**(0.00E+00)**	**(0.00E+00)**	**(0.00E+00)**
$\Delta = 50$	**0.00E+00**	**0.00E+00**	**0.00E+00**	$1.49E{-}06$
	(0.00E+00)	**(0.00E+00)**	**(0.00E+00)**	$(2.82E{-}06)$
$\Delta = 100$	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	(0.00E+00)	**(0.00E+00)**	**(0.00E+00)**	**(2.07E-08)**
$\Delta = 200$	**0.00E+00**	**0.00E+00**	**0.00E+00**	$2.56E{-}04$
	(0.00E+00)	**(0.00E+00)**	**(0.00E+00)**	$(4.77E{-}04)$
$\Delta = 500$	**0.00E+00**	**0.00E+00**	**0.00E+00**	$1.62E{-}07$
	(0.00E+00)	**(0.00E+00)**	**(0.00E+00)**	$(6.47E{-}07)$

Tableau 4.9 – Résultats obtenus sur 100 lancements en fonction de Δ et de la dimension pour une population de 300 individus, sur la fonction Griewank et avec un nombre de générations maximum égale à 5000.

Dimension	20	50	100	200
$\Delta = 10$	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	(0.00E+00)	**(0.00E+00)**	**(0.00E+00)**	**(0.00E+00)**
$\Delta = 20$	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	(0.00E+00)	**(0.00E+00)**	**(0.00E+00)**	**(0.00E+00)**
$\Delta = 50$	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	(0.00E+00)	**(0.00E+00)**	**(0.00E+00)**	**(0.00E+00)**
$\Delta = 100$	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	(0.00E+00)	**(0.00E+00)**	**(0.00E+00)**	**(0.00E+00)**
$\Delta = 200$	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	(0.00E+00)	**(0.00E+00)**	**(0.00E+00)**	**(0.00E+00)**
$\Delta = 500$	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	(0.00E+00)	**(0.00E+00)**	**(0.00E+00)**	**(0.00E+00)**

Tableau 4.10 – Résultats obtenus sur 100 lancements en fonction de Δ et de la dimension pour une population de 500 individus, sur la fonction Griewank et avec un nombre de générations maximum égale à 5000.

Dimension	20	50	100	200
$\Delta = 10$	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**
$\Delta = 20$	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**
$\Delta = 50$	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**
$\Delta = 100$	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**
$\Delta = 200$	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**
$\Delta = 500$	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**

Tableau 4.11 – Résultats obtenus sur 100 lancements en fonction de Δ et de la dimension pour une population de 800 individus, sur la fonction Griewank et avec un nombre de générations maximum égale à 5000.

Dimension	20	50	100	200
$\Delta = 10$	$5.98E{-}01$ $(3.40E{+}00)$	$9.96E{-}01$ $(6.97E{+}00)$	$9.96E{-}01$ $(9.91E{+}00)$	$1.49E{-}03$ $(1.27E{-}02)$
$\Delta = 20$	**3.99E-01** **(2.79E+00)**	$3.63E{-}07$ $(3.61E{-}06)$	**2.83E-07** **(1.64E-06)**)	**1.33E-03** **(3.78E-03)**
$\Delta = 50$	$5.98E{-}01$ $(3.40E{+}00)$	**4.37E-09** **(4.00E-08)**	$1.01E{+}00$ $(9.90E{+}00)$	$1.04E{-}01$ $(8.62E{-}01)$
$\Delta = 100$	$8.99E{-}01$ $(3.98E{+}00)$	$4.89E{-}01$ $(4.95E{+}00)$	$2.34E{+}00$ $(1.40E{+}01)$	$3.78E{+}00$ $(2.22E{+}01)$
$\Delta = 200$	$1.00E{+}00$ $(4.34E{+}00)$	$2.51E{+}00$ $(1.08E{+}01)$	$3.36E{+}00$ $(1.58E{+}01)$	$1.01E{+}00$ $(5.33E{+}00)$
$\Delta = 500$	$8.90E{+}00$ $(1.15E{+}01)$	$8.90E{+}00$ $(2.61E{+}01)$	$3.31E{+}01$ $(1.22E{+}02)$	$9.99E{+}01$ $(3.41E{+}02)$

Tableau 4.12 – Résultats obtenus sur 100 lancements en fonction de Δ et de la dimension pour une population de 50 individus, sur la fonction Rastrigin et avec un nombre de générations maximum égale à 5000.

Dimension	20	50	100	200
$\Delta = 10$	$3.98E{-}01$ $(2.79E{+}00)$	**0.00E+00** **(0.00E+00)**	$9.96E{-}01$ $(9.91E{+}00)$	**1.35E-05** **(8.40E-05)**
$\Delta = 20$	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**	$1.28E{-}03$ $(1.23E{-}02)$
$\Delta = 50$	$3.98E{-}01$ $(2.79E{+}00)$	$9.96E{-}01$ $(6.97E{+}01)$	$1.01E{+}00$ $(9.91E{+}00)$	$3.00E{-}04$ $(2.90E{-}03)$
$\Delta = 100$	$1.99E{-}01$ $(1.98E{+}00)$	$5.89E{-}06$ $(3.37E{-}05)$	$4.85E{-}04$ $(2.51E{-}03)$	$3.17E{+}00$ $(2.25E{+}01)$
$\Delta = 200$	**0.00E+00** **(2.06E-08)**	$4.98E{-}01$ $(4.95E{+}00)$	$7.06E{-}03$ $(2.46E{-}02)$	$2.18E{+}00$ $(1.98E{+}01)$
$\Delta = 500$	$2.18E{+}00$ $(6.90E{+}00)$	$4.35E{+}00$ $(1.59E{+}01)$	$1.61E{+}01$ $(4.40E{+}01)$	$4.83E{+}01$ $(1.50E{+}02)$

Tableau 4.13 – Résultats obtenus sur 100 lancements en fonction de Δ et de la dimension pour une population de 100 individus, sur la fonction Rastrigin et avec un nombre de générations maximum égale à 5000.

Dimension	20	50	100	200
$\Delta = 10$	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**
$\Delta = 20$	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(1.92E-08)**
$\Delta = 50$	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**	$8.13E{-}06$ $(7.97E{-}05)$
$\Delta = 100$	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(1.12E-08)**	$1.70E{-}05$ $(1.10E{-}04)$
$\Delta = 200$	**0.00E+00** **(1.48E-08)**	**0.00E+00** **(0.00E+00)**	$3.72E{-}06$ $(3.70E{-}05)$	$4.68E{-}04$ $(4.11E{-}03)$
$\Delta = 500$	$1.31E{-}01$ $(8.10E{-}01)$	$3.44E{+}00$ $(1.12E{+}01)$	$1.56E{+}01$ $(4.32E{+}01)$	$8.41E{+}01$ $(1.92E{+}02)$

Tableau 4.14 – Résultats obtenus sur 100 lancements en fonction de Δ et de la dimension pour une population de 300 individus, sur la fonction Rastrigin et avec un nombre de générations maximum égale à 5000.

Dimension	20	50	100	200
$\Delta = 10$	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**	**1.37E-13** **(7.78E-13)**
$\Delta = 20$	**4.38E-13** **(4.36E-12)**	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**	**1.05E-15** **(7.39E-15)**
$\Delta = 50$	**2.26E-13** **(2.24E-12)**	**3.84E-12** **(3.82E-11)**	**7.64E-14** **(7.60E-13)**	**2.88E-09** **(2.86E-08)**
$\Delta = 100$	**1.44E-09** **(1.43E-08)**	$1.90E{-}07$ $(1.37E{-}06)$	**0.00E+00** **(0.00E+00)**	**4.18E-10** **(4.03E-09)**
$\Delta = 200$	$9.03E{-}02$ $(8.98E{-}01)$	$7.84E{-}03$ $(7.77E{-}02)$	$6.59E{-}01$ $(6.51E{+}00)$	$6.65E{+}00$ $(6.48E{+}01)$
$\Delta = 500$	**1.11E-09** **(1.10E-08)**	$3.66E{-}01$ $(2.56E{+}00)$	$5.91E{-}01$ $(5.73E{+}00)$	$2.18E{+}01$ $(8.08E{+}01)$

Tableau 4.15 – Résultats obtenus sur 100 lancements en fonction de Δ et de la dimension pour une population de 500 individus, sur la fonction Rastrigin et avec un nombre de générations maximum égale à 5000.

Dimension	20	50	100	200
$\Delta = 10$	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**
$\Delta = 20$	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**
$\Delta = 50$	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**
$\Delta = 100$	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**
$\Delta = 200$	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**	$4.49E{-}01$ $(4.47E{+}00)$	$8.67E{-}05$ $(6.95E{-}04)$
$\Delta = 500$	$1.56E{-}06$ $(1.55E{-}05)$	$2.77E{-}05$ $(2.73E{-}04)$	$8.07E{-}01$ $(5.73E{+}00)$	$1.11E{+}00$ $(1.06E{+}01)$

Tableau 4.16 – Résultats obtenus sur 100 lancements en fonction de Δ et de la dimension pour une population de 800 individus, sur la fonction Rastrigin et avec un nombre de générations maximum égale à 5000.

Dimension	20	50	100	200
$\Delta = 10$	$1.15E{-}03$ $(1.15E{-}02)$	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**	$1.19E{-}03$ $(1.57E{-}03)$
$\Delta = 20$	**0.00E+00** **(0.00E+00)**	$4.27E{-}04$ $(6.65E{-}04)$	**0.00E+00** **(0.00E+00)**	$3.00E{-}04$ $(6.31E{-}04)$
$\Delta = 50$	$2.65E{-}03$ $(2.64E{-}02)$	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**	$1.52E{-}03$ $(3.30E{-}03)$
$\Delta = 100$	$1.70E{-}03$ $(1.69E{-}02)$	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**	$4.92E{-}04$ $(1.71E{-}03)$
$\Delta = 200$	$8.90E{-}03$ $(4.00E{-}02)$	$2.80E{-}06$ $(2.57E{-}05)$	$2.70E{-}05$ $(1.97E{-}04)$	$2.89E{-}02)$ $(6.34E{-}02)$
$\Delta = 500$	$3.02E{-}03$ $(1.78E{-}02)$	$7.13E{-}04$ $(2.84E{-}03)$	$4.25E{-}02$ $(1.43E{-}01)$	$3.32E{-}01$ $(3.06E{-}01)$

Tableau 4.17 – Résultats obtenus sur 100 lancements en fonction de Δ et de la dimension pour une population de 50 individus, sur la fonction Griewank et avec un nombre de générations maximum égale à 5000.

Dimension	20	50	100	200
$\Delta = 10$	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**	$1.67E{-}05$ $(2.75E{-}05)$
$\Delta = 20$	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**	$7.91E{-}05$ $(9.14E{-}05)$
$\Delta = 50$	$9.10E{-}04$ $(9.06E{-}03)$	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**	$2.45E{-}04$ $(4.26E{-}04)$
$\Delta = 100$	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**	$1.11E{-}03$ $(2.04E{-}03)$
$\Delta = 200$	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**	$1.97E{-}08$ $(6.05E{-}08)$	$1.12E{-}03$ $(5.97E{-}03)$
$\Delta = 500$	$2.95E{-}03$ $(1.51E{-}02)$	$2.73E{-}04$ $(1.94E{-}03)$	$2.36E{-}03$ $(1.69E{-}02)$	$1.18E{-}02$ $(1.05E{-}01)$

Tableau 4.18 – Résultats obtenus sur 100 lancements en fonction de Δ et de la dimension pour une population de 100 individus, sur la fonction Griewank et avec un nombre de générations maximum égale à 5000.

Dimension	20	50	100	200
$\Delta = 10$	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**
$\Delta = 20$	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**
$\Delta = 50$	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**	$1.49E{-}06$ $(2.82E{-}06)$
$\Delta = 100$	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(2.07E-08)**
$\Delta = 200$	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**	$2.56E{-}04$ $(4.77E{-}04)$
$\Delta = 500$	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**	**0.00E+00** **(0.00E+00)**	$1.62E{-}07$ $(6.47E{-}07)$

Tableau 4.19 – Résultats obtenus sur 100 lancements en fonction de Δ et de la dimension pour une population de 300 individus, sur la fonction Griewank et avec un nombre de générations maximum égale à 5000.

Dimension	20	50	100	200
$\Delta = 10$	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**
$\Delta = 20$	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**
$\Delta = 50$	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**
$\Delta = 100$	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**
$\Delta = 200$	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**
$\Delta = 500$	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**

Tableau 4.20 – Résultats obtenus sur 100 lancements en fonction de Δ et de la dimension pour une population de 500 individus, sur la fonction Griewank et avec un nombre de générations maximum égale à 5000.

Dimension	20	50	100	200
$\Delta = 10$	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**
$\Delta = 20$	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**
$\Delta = 50$	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**
$\Delta = 100$	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**
$\Delta = 200$	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**
$\Delta = 500$	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**	**0.00E+00 (0.00E+00)**

Tableau 4.21 – Résultats obtenus sur 100 lancements en fonction de Δ et de la dimension pour une population de 800 individus, sur la fonction Griewank et avec un nombre de générations maximum égale à 5000.

	Fonction	Domaine de Définition	Optimum
F_1	Sphère	$[-100, 100]^D$	0
F_2	Schwefel 2.22	$[-10,\ 10]^D$	0
F_3	Schwefel 1.2	$[-100,\ 100]^D$	0
F_4	Schwefel 2.21	$[-100,\ 100]^D$	0
F_5	Rosenbrock	$[-29,\ 31]^D$	0
F_6	Quartic	$[-1.28,\ 1.28]^D$	0
F_7	Rastrigin	$[-5.12,\ 5.12]^D$	0
F_8	Griewank	$[-600,\ 600]^D$	0
F_9	Schwefel 2.26	$[-500,\ 500]^D$	$-418.9829D$
F_{10}	Ackley	$[-32,\ 32]^D$	0
F_{11}	Shekel's Foxholes	$[-98,\ 34]^2$	$\simeq 0.998032$
F_{12}	Six-hump Camel-Back	$[-4.91017,\ 5.0893]$ $\times\ [-5.7126,\ 4.2874]$	-1.0316285
F_{13}	Branin	$[-8.142,\ 6.858]$ $\times\ [-12.275,\ 2.725]$	0.398
F_{14}	Goldstein-Price	$[-2,\ 2]\times[-3, 1]$	3
F_{15}	High conditioned elliptic	$[-100, 100]^D$	0
F_{16}	Weierstrass	$[-0.5, 0.5]^D$	0
F_{17}	Hybrid composition	$[-5, 5]^D$	0
F_{18}	Levy	$[-10, 10]^D$	0
F_{19}	Zakharov	$[-5, 10]^D$	0
F_{20}	Alpine	$[-10, 10]^D$	0
F_{21}	Pathological	$[-100, 100]^D$	0
F_{22}	Inverted cosine wave	$[-5, 5]^D$	$-D + 1$
F_{23}	Inverted cosine mixture	$[-1, 1]^D$	0
F_{24}	Epistatic Michalewicz	$[0,\ \pi]^D$	
F_{25}	Levy and Montalvo 2	$[-5, 5]^D$	0
F_{26}	Neumaier 3	$[-D^2, D^2]^D$	$-\frac{D(D+4)(D-1)}{6}$
F_{27}	Odd square	$[-15,\ 15]^D$	
F_{28}	Paviani	$[2,\ 10]^D$	
F_{29}	Periodic	$[-10, 10]^D$	0.9
F_{30}	Salomon	$[-100, 100]^D$	0
F_{31}	Shubert	$[-10, 10]^D$	$\simeq -186.7309$
F_{32}	Sinusoidal	$[0, 180]^D$	$-A - 1$ avec $A = 2.5$
F_{33}	Michalewicz	$[0,\pi]^D$	
F_{34}	Whitely	$[-100,\ 100]^D$	

Tableau 4.22 – Problèmes d'optimisation F_1-F_{28}. D est la dimension.

	Fonction	Définition
F_1	Sphère	$\sum_{i=1}^{D} x_i^2$
F_2	Schwefel 2.22	$\sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert$
F_3	Schwefel 1.2	$\sum_{i=1}^{D} (\sum_{j=1}^{D} x_j)^2$
F_4	Schwefel 2.21	$\max \{\lvert x_i \rvert, 1 \leq i \leq D\}$
F_5	Rosenbrock	$\sum_{i=1}^{D-1} (100(x_i^2 - x_{i+1})^2 + (x_i - 1^2))$
F_6	Quartic	$\sum_{i=1}^{D} i x_i^4 + random[0,1]$
F_7	Rastrigin	$\sum_{i=1}^{D} (x_i^2 - 10\cos(2\pi x_i) + 10)$
F_8	Griewank	$\sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{D} \cos(\frac{x_i}{\sqrt{i}}) + 1$
F_9	Schwefel 2.26	$-\sum_{i=1}^{D} (x_i \sin \sqrt{\lvert x_i \rvert})$
F_{10}	Ackley	$-20\exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2})$ $-\exp(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)) + 20$ $+e$
F_{11}	Shekel's Foxholes	$[\frac{1}{500} + \sum_{i=1}^{25} \frac{1}{i + \sum_{j=1}^{2} (x_j - a_{j,i})^6}]^{-1}$
F_{12}	Six-hump Camel-Back	$4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2$ $+4x_2^4$
F_{13}	Branin	$(x_2 - \frac{5}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6)^2 + 10(1 - \frac{1}{8\pi})\cos x_1 + 10$
F_{14}	Goldstein-Price	$g(x) \times h(x)$ avec $g(x) = 1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)$ $h(x) = 30 + (2x_1 - 3x_2)^2(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)$
F_{15}	High conditioned elliptic	$\sum_{i=1}^{D} (10^{\frac{6(i-1)}{D-1}} x_i^2)$
F_{16}	Weierstrass	$\sum_{i=1}^{D} (\sum_{j=0}^{D} (a^j \cos 2\pi b^j (x_i - 0.5)) - D \sum_{j=1}^{D} a^j \cos \pi b^k)$ avec $a = 0.5, b = 3$

Tableau 4.23 – Fonctions $F_1 - F_{16}$

	Fonction	Définition
F_{17}	Hybrid composition	$\sum_{i=1}^{10} w_i(g_i(\lambda_i(x-o_i))+b_i)$ avec $o_i = o_{i,j}$, $C = 2000$ $g_i(\lambda_i x) = C \times \frac{h_i(\lambda_i x)}{h_i(\lambda_i[5,...,5])}$ $h_{1-2} = F_7(x)$, $h_{3-4} = F_{16}(x)$, $h_{5-6} = F_8(x)$ $h_{7-8} = F_{10}(x)$, $h_{9-10} = F_1(x)$ $\omega = \exp(-\sum_{j=i}^{D} \frac{(x_j-o_{i,j})^2}{2D\sigma_j^2})$ $\sigma_j = 1$ pour $j = 1, 2, ..., D$ $\lambda_i = [1, 1, 0.1, 0.1, 12, 12, 6.4, 6.4, 20, 20]$ $b_i = [0, 100, 200, 300, 400, 500, 600, 700, 800, 900]$ $o_i = [u_i, u_i, ..., u_i]$ avec $u_i = 0.5 \times floor(i/2)$
F_{18}	Levy	$\sin^2(\pi x_1) + \sum_{i=1}^{D-1}((x_i-1)^2(1+10\sin^2(\pi x_i+1)))$ $+(x_n^2-1)^2(1+10\sin^2 \pi x_n)$
F_{19}	Zakharov	$\sum_{i=1}^{D} x_i^2 + (\sum_{i=1}^{D} 0.5ix_i)^2 + (\sum_{i=1}^{D} 0.5ix_i)^4$
F_{20}	Alpine	$\sum_{i=1}^{D} \lvert x_i \sin(x_i) + 0.1x_i \rvert$
F_{21}	Pathological	$\sum_{i=1}^{D-1} 0.5 + \frac{\sin^2\sqrt{100x_i^2+x_{i+1}^2}-0.5}{1+0.001(x_i^2-2x_ix_{i+1}+x_{i+1}^2)^2}$
F_{22}	Inverted cosine wave	$-\sum_{i=1}^{D-1} \exp\frac{-(x_i^2+x_{i+1}^2+0.5x_ix_{i+1})}{8}$ $\times \cos(4\sqrt{x_i^2+x_{i+1}^2+0.5x_ix_{i+1}})$
F_{23}	Inverted cosine mixture	$0.1D - (0.1\sum_{i=1}^{D}\cos 5\pi x_i + \sum_{i=1}^{D} x_i^2)$
F_{24}	Epistatic Michalewicz	$-\sum_{i=1}^{D} \sin y_i^2 \sin^{2m}(\frac{iy_i^2}{\pi})$ avec $y_i = x_i\cos\theta - x_{i+1}\sin\theta$ for i= $1, 3, 5, ... < D; y_i = x_i\sin\theta + x_{i+1}\cos\theta$ pour $i = 2, 4, 6, ... < D; y_i = x_i$ pour $i = D$
F_{25}	Levy and Montalvo 2	$0.1(\sin^2(3\pi x_1) + \sum_{i=1}^{D-1}(x_i-1)^2(1+10\sin^2(3\pi x_{i+1}))$ $+(x_n-1)^2(1+10\sin^2(2\pi x_n)))$
F_{26}	Neumaier 3	$\sum_{i=1}^{D}(x_i-1)^2 - \sum_{i=2}^{D} x_ix_{i-1}$
F_{27}	Odd square	$-(1+\frac{0.2n}{N+0.01})\cos(N\pi)\exp-\frac{N}{2\pi}$ avec $N = \sqrt{D}\max_{1\le i\le D}\lvert x_i - b_i\rvert$, $n = \sqrt{\sum_{i=1}^{D}(x_i-b_i)^2}$

Tableau 4.24 – Fonctions $F_{17} - F_{30}$

	Fonction	Définition
F_{28}	Paviani	$\sum_{i=1}^{D}[(ln(x_i-2))^2+(ln(10-x_i))^2]-(\prod_{i=1}^{D}x_i)^{0.2}$
F_{29}	Periodic	$1+\sum_{i=1}^{D}\sin^2(x_i)-0.1\prod_{i=1}^{D}\exp(-x_i^2)$
F_{30}	Salomon	$1-\cos(2\pi\sqrt{\sum_{i=1}^{D}x_i^2})+0.1\sqrt{\sum_{i=1}^{D}x_i^2}$
F_{31}	Shubert	$\prod_{i=0}^{D}(\sum_{j=1}^{5}j\cos((j+1)x_i+j))$
F_{32}	Sinusoidal	$-[A\prod_{i=1}^{D}\sin\frac{(x_i-z)\pi}{180}+\prod_{i=1}^{D}\sin(\frac{B(x_i-z)\pi}{180})]$ avec $A=2.5, B=5, z=30$
F_{33}	Michalewicz	$-\sum_{i=1}^{D}(sin(x_i^2)sin^{2m}(\frac{ix_i^2}{\pi})$ avec $m=10$
F_{34}	Whitely	$\sum_{j=1}^{D}\sum_{i=1}^{D}(\frac{y_{i,j}}{4000}-\cos y_{i,j}+1)$ avec $y_{i,j}=100(x_j-x_i^2)^2+(1-x_i)^2$

Tableau 4.25 – Fonctions $F_{28}-F_{34}$

Fonction	F_1		F_2		F_3		F_4	
Dimension	30	40	30	40	30	40	30	40
$HdEA$								
Moy	**0.00**	**0.00**	**0.00**	0.0034	16920.23	34006.13	10.8802	22.6509
Std	**0.00**	**0.00**	**0.00**	0.0013	2818.22	4440.54	1.3212	1.8335
$RCGA-UNDX$								
Moy	**0.00**	**0.00**	**0.00**	0.0005	2811.5869	27142.12	91.8556	99.9358
Std	**0.00**	**0.00**	**0.00**	0.0002	1668.95	7377.41	20.8748	0.352
$CMA-ES$								
Moy	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
Std	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**
DE								
Moy	0.0221	0.167	0.3594	1.1491	26073.04	45895.90	49.5082	60.7734
Std	0.0048	0.0279	0.0371	0.1101	3339.06	4761.74	4.0114	3.8266
ODE								
Moy	0.000006	0.001	0.0953	0.3459	78.1691	409.988	0.0111	0.1528
Std	0.00	0.0009	0.0486	0.1754	45.1045	230.1221	0.0808	0.4848
$DEahcSPX$								
Moy	0.1075	0.0071	0.0322	0.0011	65.9908	310.1678	15.5882	15.8102
Std	0.399	0.0389	0.1423	0.0045	65.222	161.2674	3.5293	3.6463
$DPSO$								
Moy	3.3462	6.9563	8.8458	19.4899	1955.2153	3986.0922	10.539	13.2989
Std	0.9949	1.2491	1.1698	5.3605	24.5853	35.7403	1.3743	1.4465
$SEPSO$								
Moy	2.8527	6.8806	8.8184	21.4078	2984.5912	6057.7894	13.867	16.9939
Std	0.9324	1.5195	1.398	5.7894	29.6756	41.8452	1.7487	1.663
EDA								
Moy	3439.532	5203.446	22.252	34.532	3749.133	7174.429	21.142	25.884
Std	1221.21	1828.79	5.433797	7.02534	1294.74	3081.31	5.932983	6.132852
$CoDE$								
Moy	0.00002	0.0008	0.0016	0.01417	15.3733	166.1292	0.3477	1.5189
Std	0.00001	0.0005	0.0006	0.00439	9.7621	81.4287	0.1162	0.8251
$JADE$								
Moy	**0.00**	**0.00**	**0.00**	0.00001	0.3452	21.0160	0.0305	0.7380
Std	**0.00**	**0.00**	**0.00**	0.00001	0.4739	18.1420	0.0308	0.3968
jDE								
Moy	0.00009	0.0051	0.0011	0.01177	372.3809	1590.7465	2.6356	7.3268
Std	0.00005	0.0028	0.0004	0.00313	175.2664	672.0046	1.6114	2.6444
$MPEDE$								
Moy	0.00690	0.0633	0.0909	0.26141	13.9150	106.5200	0.6073	1.5490
Std	0.00264	0.0219	0.0269	0.06018	9.6290	51.7790	0.1502	0.3240
MEA								
Moy	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	0.0007	0.0052	0.0050
Std	**0.00**	**0.00**	**0.00**	**0.00**	**0.00**	0.0065	0.01185	0.0151
$EAOG$								
Moy	**0.00**	**0.00**	**0.00**	0.0671	0.0001	**0.00**	0.0009	**0.00**
Std	**0.00**	**0.0004**	**0.00**	19.2769	0.0994	**0.0004**	0.0875	**0.3932**

Tableau 4.26 – Résultats expérimentaux obtenus pour les fonctions $F_1 - F_4$

Fonction	F_5		F_6		F_7		F_8	
Dimension	30	40	30	40	30	40	30	40
$HdEA$								
Moy	21.1276	93.6767	10.4615	16.1244	**0.00**	0.0082	**0.00**	0.001
Std	13.7111	27.2255	0.6028	0.8751	**0.00**	0.0048	**0.00**	0.0026
$RCGA-UNDX$								
Moy	71.1275	78.3633	8.8836	13.2587	219.8173	340.1991	1.6946	11.3365
Std	67.2881	72.0327	0.4382	0.5691	12.6701	14.534	0.1151	1.5104
$CMA-ES$								
Moy	**0.00**	**0.00**	0.2303	0.2659	53.6481	76.1739	0.0014	0.0012
Std	**0.00**	**0.00**	0.0893	0.0963	14.1662	17.1046	0.0036	0.0031
DE								
Moy	614.4588	2023.166	10.9846	16.3383	25.7105	57.3944	0.9931	1.1542
Std	112.1748	405.6226	0.5641	0.6334	2.65	5.3769	0.0301	0.0259
ODE								
Moy	27.0344	42.2877	8.5199	12.4331	102.3277	170.7193	0.0258	0.2427
Std	0.7121	13.7948	0.4121	0.5279	37.9763	55.5588	0.0326	0.1488
$DEahcSPX$								
Moy	3160.5891	2287.669	8.7536	13.0129	40.807	189.1699	0.1613	0.039
Std	9387.60	9595.24	0.52	0.5081	26.7653	47.6562	0.2685	0.0604
$DPSO$								
Moy	12789.49	31765.02	9.9202	14.6241	125.4958	201.5571	4.0567	7.0049
Std	78.0953	110.4379	0.8192	0.8821	3.9027	4.6532	0.8773	1.1996
$SEPSO$								
Moy	10464.07	41826.01	10.6244	15.6301	112.9555	181.948	3.6995	7.433
Std	80.0175	141.3864	0.8755	1.0337	4.5789	4.8482	0.9511	1.2926
EDA								
Moy	30214.733	61973.272	100.069	353.440	188.384	272.859	30.504	49.551
Std	12162.08	25541.42	52.002	142.018	20.555	29.016	10.987	17.855
$CoDE$								
Moy	28.0896	47.6793	0.0204	0.0324	60.6495	133.1235	**0.00**	**0.00**
Std	11.7630	21.6289	0.0068	0.0105	7.9473	13.2087	**0.00**	**0.00**
$JADE$								
Moy	22.1020	37.2830	0.0066	0.0100	21.4990	54.3660	**0.00**	**0.00**
Std	6.1302	12.9880	0.0021	0.0029	1.8989	4.7659	**0.00**	**0.00**
jDE								
Moy	41.5591	108.4968	0.0285	0.0499	30.2820	67.3089	**0.00**	**0.00**
Std	26.3418	58.2434	0.0073	0.0103	4.9835	7.4604	**0.00**	**0.00**
$MPEDE$								
Moy	30.0190	54.4020	0.0094	0.0136	89.3420	149.3000	**0.00**	**0.00**
Std	11.0600	28.1810	0.0026	0.0036	11.6390	15.7860	**0.00**	**0.00**
MEA								
Moy	6.6140	5.6453	0.0080	0.0084	**0.00**	0.0141	**0.00**	**0.00**
Std	18.1819	23.422	0.0049	0.0058	**0.0470**	0.0664	**0.00**	**0.00**
$EAOG$								
Moy	2.3123	12.8736	**0.0002**	**0.0003**	**0.00**	**0.00**	**0.00**	**0.00**
Std	8.2330	19.9894	**0.0002**	**0.0006**	**2.8579**	**5.4082**	**0.2619**	**0.0457**

Tableau 4.27 – Résultats expérimentaux obtenus pour les fonctions $F_5 - F_8$

Fonction	F_9		F_{10}		F_{11}	F_{12}	F_{13}	F_{14}
Dimension	30	40	30	40	2	2	2	2
$HdEA$								
Moy	**-13780.72**	**-18374.62**	**0.00**	0.005	1.2082	**-1.0316**	0.401	4.4101
Std	**25.3953**	**18.2885**	**0.00**	0.002	0.7333	**0.0001**	0.0264	4.0821
$RCGA-UNDX$								
Moy	−5941.04	−6580.023	20.7175	20.8336	6.5167	−0.6587	0.4596	56.0632
Std	486.7029	470.1085	0.0904	0.1001	3.1656	0.3117	0.0565	29.5616
$CMA-ES$								
Moy	−5406.98	−7187.4	21.3439	21.495	13.5224	−1.0235	0.3979	7.32
Std	95.6	184.1	0.4316	0.1658	5.3565	0.0816	0.00	16.6041
DE								
Moy	−12801.84	−15568.76	1.8269	3.047	13.9738	−0.6695	1.5219	14.8393
Std	156.2624	256.1296	0.3319	0.2579	21.6527	0.3211	1.348	10.4039
ODE								
Moy	−5475.53	−6559.977	0.0099	0.1181	2.4491	−1.0214	0.425	3.5207
Std	506.0389	839.0191	0.0117	0.1698	1.4992	0.0109	0.0277	0.4767
$DEahcSPX$								
Moy	−10285.82	−9462.16	2.8201	4.6082	19.077	−0.4882	6.683	21.806
Std	692.3803	1390.58	4.0335	5.1296	61.8988	3.2941	19.2551	85.0168
$DPSO$								
Moy	−5185.645	−6196.144	6.0714	6.9424	1.491	−1.0229	1.4407	3.1429
Std	25.7171	27.7505	0.821	0.8013	0.84	0.1171	1.6188	0.6474
$SEPSO$								
Moy	−7702.762	−9108.41	6.4363	7.7494	2.2467	−1.0252	0.4096	3.0618
Std	27.2504	30.4254	1.0018	0.9357	1.1537	0.104	0.1366	0.2934
EDA								
Moy	−4679.949	−5510.891	10.177	11.035	2.821	−1.031	0.398	**3.000**
Std	703.073	742.236	1.29405	1.3213	1.7371	0.0012	0	**0.00**
$CoDE$								
Moy	−12186.08	−13754.91	**0.00**	0.01	1.5878	−1.0298	0.9835	3.0975
Std	292.25	467.30	**0.00**	0.00	0.7890	0.0018	0.0006	0.1486
$JADE$								
Moy	−12076.00	−14445.00	**0.00**	**0.00**	1.8799	−1.0267	0.40003	3.1321
Std	139.26	292.99	**0.00**	**0.00**	1.0806	0.0048	0.0022	0.1397
jDE								
Moy	−12076.00	−14445.00	**0.00**	**0.00**	2.5218	−1.0255	0.4083	3.4379
Std	139.26	292.99	**0.00**	**0.00**	1.4111	0.0066	0.0139	0.4622
$MPEDE$								
Moy	−12076.00	−14445.00	**0.00**	**0.00**	2.1429	−1.0020	0.4277	3.5707
Std	139.26	292.99	**0.00**	**0.00**	1.1342	0.0278	0.0327	0.6409
MEA								
Moy	−12427.4	−16759.3	**0.00**	0.0001	**0.8500**	−1.0313	**0.3979**	3.0007
Std	696.272	0.0000	**0.00**	**0.0000**	**2.6847**	0.0004	**0.0000**	0.0036
$EAOG$								
Moy	−8108.8820	$-9.52E{+}03$	**0.00**	**0.00**	2.6160	−1.0236	0.4019	3.9036
Std	1129.7026	1396.4631	**0.3744**	**0.1261**	2.8421	0.0097	0.0057	1.2277

Tableau 4.28 – Résultats expérimentaux obtenus pour les fonctions $F_9 - F_{14}$

Fonction	F_{15}		F_{16}		F_{17}		F_{18}	
Dimension	30	40	30	40	30	40	30	40
$HdEA$								
Moy	**0.00**	1.6503	0.0047	0.1451	8814.832	14179.81	**0.00**	**0.00**
Std	**0.00**	2.7983	0.0014	0.0286	2547.73	4285.01	**0.00**	**0.00**
$RCGA-UNDX$								
Moy	69184345	$1.67E{+}8$	30.1414	65.2566	27770.55	34420.03	0.3329	8.5853
Std	$1.2E{+}7$	$2.8E{+}7$	6.7271	2.2689	4571.35	6050.13	0.0954	1.8914
$CMA-ES$								
Moy	41.4371	116566.8	7.1621	10.5753	6410.02	6698.98	0.0115	**0.00**
Std	100.7638	76286.01	12.9456	18.654	362.67	324.05	0.1153	**0.00**
DE								
Moy	4659.844	37580.67	1.485	3.3923	2335.616	2454.875	0.0032	0.0278
Std	926.189	6525.303	0.119	0.2111	83.3354	77.024	0.0008	0.0059
ODE								
Moy	1.1579	254.327	2.1745	4.2087	1851.785	2003.177	0.000015	0.0006
Std	1.2544	302.6105	0.4339	0.8853	164.0192	163.5926	0.00	0.0005
$DEahcSPX$								
Moy	38175.4	2085.96	0.7444	0.64	2637.816	2665.809	0.0598	0.0616
Std	200625.0	10954.24	0.6961	0.6938	113.3806	93.9438	0.1701	0.2015
$DPSO$								
Moy	8264438	15086215	13.0929	19.525	10080636	11335533	4.1547	7.0132
Std	1925.56	2359.02	1.2306	1.3333	1168.58	1009.98	1.6574	1.8273
$SEPSO$								
Moy	6281119	13793165	14.0028	20.9514	234963	365663.1	3.6763	7.4596
Std	1550.51	2260.61	1.3528	1.5315	277.2721	398.9803	1.4536	1.886
EDA								
Moy	6751507.8	23576245	16.726	24.507	74236.525	$2.599E{+}9$	5.980	12.734
Std	4130691	$1.19E{+}7$	2.452092	3.438128	55975.6	$2.5E{+}10$	3.278065	6.62116
$CoDE$								
Moy	0.0296	1.7310	0.0968	0.3601	7105.60	9478.7652	0.0000011	0.0055
Std	0.0207	1.0660	0.0215	0.0779	0.0437	1.1007	0.000001	0.0462
$JADE$								
Moy	**0.00**	**0.00**	0.0096	0.0169	7105.40	9474.1000	0.000895	0.0081
Std	**0.00**	**0.00**	0.0046	0.0081	0.0010	0.0473	0.008953	0.0484
jDE								
Moy	0.2795	13.9615	0.0409	0.2039	26049319.52	25967654.01	0.000002	0.0001
Std	0.1397	5.7529	0.0079	0.0368	5569368.33	6133867.50	0.000001	0.00005
$MPEDE$								
Moy	224.2200	2050.3000	0.0057	0.0483	889870.00	1810700.00	0.000330	0.0083
Std	94.1680	879.6600	0.0023	0.0206	262030.00	434160.00	0.000154	0.0229
MEA								
Moy	0.0010	0.0062	**0.0027**	0.0044	**0.0203**	**0.0120**	**0.00**	**0.00**
Std	0.0059	0.0281	**0.0147**	0.0219	**0.4952**	**0.0023**	**0.00**	**0.00**
$EAOG$								
Moy	0.0009	0.0016	0.1664	**0.0001**	11.4881	11.1367	0.0003	0.2847
Std	0.0086	0.0431	2.8132	**5.4040**	1.2555	0.6243	0.0026	2.7775

Tableau 4.29 – Résultats expérimentaux obtenus pour les fonctions $F_{15} - F_{18}$

Fonction	F_{19}		F_{20}		F_{21}		F_{22}	
Dimension	30	40	30	40	30	40	30	40
$HdEA$								
Moy	261.3953	395.2277	0.0004	0.0057	4.8663	7.6184	−24.9443	−31.9794
Std	34.998	42.3205	0.0002	0.001	0.3451	0.399	0.9411	0.9206
$RCGA-UNDX$								
Moy	293.5157	496.9959	10.2837	27.5203	7.61	11.0298	−6.633	−7.12
Std	36.933	45.4164	1.3909	1.9434	0.5734	0.7434	0.452	0.6458
$CMA-ES$								
Moy	**0.00**	**0.00**	0.0025	0.0097	13.7823	18.9196	−0.9678	−1.0698
Std	**0.00**	**0.00**	0.0028	0.0079	0.2792	0.2544	0.732	0.7926
DE								
Moy	274.0171	425.7634	0.1641	1.1909	5.3987	8.2231	−18.8816	−22.652
Std	30.0291	42.4568	0.0486	0.2507	0.5198	0.5673	0.556	0.7497
ODE								
Moy	24.2815	164.5583	0.0299	0.0865	0.0237	0.1127	−27.8856	−37.6543
Std	8.0691	38.622	0.0099	0.0887	0.1431	0.3907	1.8404	2.359
$DEahcSPX$								
Moy	2.2143	30.1464	0.0013	**0.0001**	4.6963	6.1197	−14.6745	−11.075
Std	3.8524	13.0009	0.0078	**0.0003**	1.3226	2.1082	4.0927	1.9585
$DPSO$								
Moy	134.6547	278.6536	5.758	9.024	11.8132	16.3968	−15.4114	−18.3188
Std	7.912	10.7637	1.3801	1.5181	0.521	0.5582	1.2455	1.382
$SEPSO$								
Moy	71.7805	155.882	9.5052	14.9113	12.0147	16.7829	−16.9436	−20.5577
Std	4.8726	6.205	1.7957	1.9475	0.532	0.5337	1.2194	1.3924
EDA								
Moy	69.740	94.707	12.235	20.463	12.309	17.025	−18.728	−22.469
Std	29.0711	32.8311	3.110696	4.22087	0.17794	0.212107	4.185106	4.879363
$CoDE$								
Moy	0.0657	1.6293	4.7025	9.3020	14.5000	19.5001	−19.8109	−15.8452
Std	0.0470	1.0418	2.0291	4.8360	0.0000	0.0000	2.2239	2.1074
$JADE$								
Moy	1.0921	12.3280	0.0249	0.0408	14.5000	19.5000	−15.8210	−14.3610
Std	3.6314	22.4920	0.0128	0.0314	0.0000	0.0000	1.2548	0.8821
jDE								
Moy	21.6786	144.8795	0.0114	0.0282	14.5000	19.5000	−19.0921	−16.1965
Std	9.8599	40.8360	0.0019	0.0054	0.0000	0.0000	1.7091	1.6248
$MPEDE$								
Moy	0.7871	5.8333	0.0302	0.0434	14.5000	19.5000	**-29**	**-39**
Std	0.4612	2.5585	0.0026	0.0038	0.0000	0.0001	**0.00001**	**0.0001**
MEA								
Moy	0.7153	5.0610	0.0001	0.0002	0.0883	0.2740	−25.0738	−30.3853
Std	3.57197	13.8723	0.0003	0.0012	0.4948	0.9273	5.2767	7.5871
$EAOG$								
Moy	0.1094	0.0311	**0.00**	0.0002	**0.0032**	**0.0671**	−26.2906	**-39**
Std	80.9435	147.7874	**0.2857**	0.4852	**0.9421**	**1.0758**	6.0365	**0.0001**

Tableau 4.30 – Résultats expérimentaux obtenus pour les fonctions $F_{19} - F_{22}$

Fonction	F_{23}		F_{24}		F_{25}		F_{26}	
Dimension	30	40	30	40	30	40	30	40
$HdEA$								
Moy	**0.00**	**0.00**	**-25.3678**	**-32.2103**	0.1626	2.4995	8025.425	73996.06
Std	**0.00**	**0.00**	**0.572**	**0.6295**	0.4616	4.3733	4773.20	24998.18
$RCGA-UNDX$								
Moy	0.3566	2.4797	−8.8451	−10.32	152.9672	566.6314	62837.03	296115.4
Std	0.073	0.1908	0.5691	0.5004	17.6976	81.3621	3012.70	17576.71
$CMA-ES$								
Moy	0.4493	0.6355	−19.1834	−24.0149	0.023	0.0219	−2428.19	−413.708
Std	0.258	0.2534	1.8797	1.8759	0.0472	0.0465	0.00	0.00
DE								
Moy	0.0027	0.0189	−18.3183	−22.3111	60.0966	131.245	122598.2	886175.2
Std	0.0006	0.0031	0.6445	0.6373	10.9953	10.733	25422.81	139048.9
ODE								
Moy	0.000027	0.0010	−12.5543	−14.4373	26.0994	47.3521	−4930	11440
Std	0.00	0.0008	1.2739	1.3917	12.9456	15.3637	1162.05	3206.49
$DEahcSPX$								
Moy	0.1752	0.1894	−12.9365	−13.2637	37.1675	52.3789	1911.297	8562.511
Std	0.1499	0.1578	2.0401	−1.3542	17.6322	19.5292	4085.18	9956.65
$DPSO$								
Moy	0.6795	1.1519	−10.3292	−12.0257	135.0221	245.9962	26342.66	154356.6
Std	0.4754	0.5166	0.8904	0.9585	6.6864	8.3421	86.3998	205.9114
$SEPSO$								
Moy	0.7684	1.3428	−10.9954	−12.5813	152.5561	259.8431	30572.09	165457.5
Std	0.5383	0.5792	0.9763	1.0432	7.1358	8.0873	100.8033	229.2899
EDA								
Moy	1.885	2.979	−9.361	−11.071	10.527	18.692	141156.77	835826.06
Std	0.444738	0.553565	0.75983	0.888224	4.54527	5.99804	65517.19	326503.5
$CoDE$								
Moy	**0.00**	**0.00**	−15.72	−18.42	**0.00**	0.00011	$3.35E+08$	$1.08E+10$
Std	**0.00**	**0.00**	0.68	0.70	**0.00**	0.00111	$4.12E+08$	$9.15E+09$
$JADE$								
Moy	**0.00**	**0.00**	−18.93	−22.95	**0.00**	**0.00**	$1.67E+08$	$4.79E+09$
Std	**0.00**	**0.00**	0.57	0.76	**0.00**	**0.00**	$1.77E+08$	$3.70E+09$
jDE								
Moy	**0.00**	**0.00**	−18.38	−22.43	**0.00**	0.000002	$1.97E+09$	$5.42E+10$
Std	**0.00**	**0.00**	0.71	0.80	**0.00**	0.000001	$1.50E+09$	$2.84E+10$
$MPEDE$								
Moy	**0.00**	0.0002	−15.61	−18.76	0.00002	0.00009	−3329.30	−787.56
Std	**0.00**	0.0001	0.89	1.06	0.000011	0.00006	845.60	2818.50
MEA								
Moy	**0.00**	**0.00**	−15.7793	−18.5887	**0.00**	**0.00**	**-4287.36**	**-8056.9000**
Std	**0.00**	**0.00**	1.6019	2.14901	**0.00**	**0.00**	**261.43**	**774.064**
$EAOG$								
Moy	**0.00**	**0.00**	−19.2315	−22.7894	0.0380	2.7605	$1.09E{+}11$	$1.99E{+}12$
Std	**0.00**	**0.00**	3.7626	4.7421	0.2183	0.0255	251.0113	465.4929

Tableau 4.31 – Résultats expérimentaux obtenus pour les fonctions $F_{23} - F_{26}$

Fonction	F_{27}		F_{28}		F_{29}		F_{30}	
Dimension	30	40	30	40	30	40	30	40
$HdEA$								
Moy	-0.0004	-0.0001	-997867	**-1E+8**	1.0004	1.0046	1.2051	2.1347
Std	0.001	0.0006	0.0271	**0.00**	0.0002	0.0012	0.1365	0.2002
$RCGA-UNDX$								
Moy	-0.0001	0.00	-330341	-509252	6.9638	9.9936	2.3127	5.0289
Std	0.00	0.00	0.00	0.00	0.3607	0.4529	0.1817	0.397
$CMA-ES$								
Moy	-0.0107	-0.0032	-951804	$-9.8E{+}7$	8.142	13.0446	1.1979	1.4629
Std	0.0195	0.0064	187570.7	4361415	5.7645	7.2567	0.247	0.2635
DE								
Moy	-0.0001	0.00	-958473	$-8.8E{+}7$	1.4523	2.1431	3.6819	5.476
Std	0.0001	0.00	6695.09	1605337	0.076	0.1473	0.2572	0.3649
ODE								
Moy	-0.8233	-0.7227	-610112	$-3.8E{+}07$	0.9107	0.9137	0.4718	0.9017
Std	0.058	0.1171	37311.6	3450073	0.0418	0.0581	0.1056	0.1632
$DEahcSPX$								
Moy	-0.0416	-0.0265	-996116	$-9.5E{+}7$	2.4177	5.8216	0.4953	0.5465
Std	0.0311	0.0235	5578.90	1488854	0.8642	1.16	0.1756	0.1399
$DPSO$								
Moy	-0.051	-0.0192	-342933	$-1.7E{+}07$	4.8472	7.7492	2.9157	3.8791
Std	0.1507	0.094	217.3297	2125.222	0.8796	0.9589	0.6078	0.629
$SEPSO$								
Moy	-0.0272	-0.0101	-965431	$-8.8E{+}07$	3.3681	5.5601	3.3948	4.6228
Std	0.1234	0.0912	24.4007	3225.70	0.7988	0.9266	0.7648	0.7211
EDA								
Moy	-0.060	-0.018	-286765.1	-14874856	7.629	11.228	5.186	6.780
Std	0.04363	0.014031	36881.35	2468007	0.544293	0.552775	1.072429	1.362282
$CoDE$								
Moy	0.00	0.00	**-999394**	-99975946	1.7532	2.8483	0.4583	0.7648
Std	$4.40E-10$	$9.02E-14$	**0.15**	4780.38	0.1252	0.2612	0.0616	0.1017
$JADE$								
Moy	0.00	0.00	-999390	-99997000	1.1813	1.5935	0.2187	0.2936
Std	$8.08E-25$	$7.93E-36$	0.0001	98.84	0.0331	0.0870	0.0365	0.0242
jDE								
Moy	0.00	0.00	-999392	-99970459	1.3588	1.9327	0.5940	0.9490
Std	$1.793E-25$	$2.26E-37$	0.69	4637.86	0.0873	0.1841	0.0662	0.0989
$MPEDE$								
Moy	-0.000001	0.00	-999270	-99689000	2.3531	3.7370	0.5215	0.7863
Std	$3.35E-06$	$2.32E-08$	41.665	59036	0.3047	0.4921	0.0591	0.0817
MEA								
Moy	-0.1680	-0.1249	-997867	**-1.0E+08**	0.9300	0.9394	0.1249	0.1350
Std	0.0697	0.0039	0.0000	**0.00**	0.0458	0.0485	0.0572	0.0517
$EAOG$								
Moy	**-0.9941**	**-0.9909**	-982378	$-9.43E{+}07$	**0.9025**	**0.9001**	**0.0070**	**0.1141**
Std	**0.0142**	**0.0213**	0.7646	0.1796	**0.0176**	**0.0140**	**0.0353**	**0.4365**

Tableau 4.32 – Résultats expérimentaux obtenus pour les fonctions $F_{27} - F_{30}$

Fonction	F_{31}		F_{32}		F_{33}		F_{34}	
Dimension	30	40	30	40	30	40	30	40
$HdEA$								
Moy	$-2.0E{+}34$	$-2.7E{+}45$	-1.521	-1.3187	**-29.559**	**-38.8299**	20.6012	694.993
Std	$3.6E{+}33$	$6.7E{+}44$	0.6748	0.5164	**0.0289**	**0.1222**	28.8726	145.0827
$RCGA-UNDX$								
Moy	$-1.3E{+}20$	$-3.7E{+}25$	-3.3678	-2.4558	-10.2002	-12.0481	7440.466	722794.5
Std	$2.1E{+}20$	$9.9E{+}25$	0.0231	0.0556	0.5695	0.5948	2193.41	204048.1
$CMA-ES$								
Moy	$-1.1E{+}29$	$-1.5E{+}36$	-2.6016	-2.52	-19.1408	-24.5539	319.3721	541.3366
Std	$4.4E{+}29$	$8.1E{+}36$	1.5276	1.5794	2.0299	2.7287	102.4076	178.4243
DE								
Moy	$-9E{+}29$	$-1.6E{+}38$	-2.1663	-1.5323	-24.8678	-30.0203	830.2062	1765.99
Std	$1.6E{+}30$	$3.5E{+}38$	0.1757	0.237	0.418	0.5516	15.6812	56.5738
ODE								
Moy	$-1.2E{+}24$	$-3.7E{+}31$	**-3.5**	**-3.4892**	-14.7206	-17.6109	766.9481	1438.332
Std	$7.1E{+}24$	$2.9E{+}32$	**0.00**	**0.0412**	1.0599	1.357	22.6609	26.614
$DEahcSPX$								
Moy	$-1.9E{+}24$	$-3.1E{+}29$	-3.3078	-3.479	-16.9775	-15.5823	9536.837	2341.138
Std	$1.2E{+}25$	$3.0E{+}30$	0.3708	0.1145	2.3975	1.7578	38054.29	4702.41
$DPSO$								
Moy	$-2.7E{+}34$	$-6.2E{+}31$	-2.3102	-1.8404	-13.6114	-15.8861	11716.56	45903.7
Std	$4.1E{+}12$	$8.0E{+}15$	0.7575	0.7312	0.965	1.0123	77.5304	147.5638
$SEPSO$								
Moy	$-2.1E{+}25$	$-6.2E{+}31$	-2.3102	-1.8404	-14.0837	-16.6464	11007.86	56578.07
Std	$8.6E{+}12$	$1.8E{+}16$	0.8684	0.8695	1.1034	1.1435	81.4433	176.2171
EDA								
Moy	$-1.2E{+}22$	$-3.01E{+}29$	-1.222	-0.548	-10.781	-12.962	985056.31	2262390.1
Std	$8.9E{+}22$	$2.4E{+}30$	0.287902	0.195443	0.708519	1.195155	769131.6	1513021
$CoDE$								
Moy	$-3.32E+29$	$-3.78E+36$	-1.5612	-1.4484	-15.7670	-18.2819	708.1563	1364.7692
Std	$5.68E+29$	$9.09E+36$	0.1159	0.1824	0.6913	0.6640	24.1381	29.4917
$JADE$								
Moy	$-1.59E+30$	$-1.04E+38$	-1.6249	-1.5017	-18.8660	-22.7770	537.3400	1107.7000
Std	$1.26E+30$	$2.39E+38$	0.0461	0.1007	0.5778	0.6310	24.6780	39.4210
jDE								
Moy	$-3.88E+30$	$-8.70E+38$	-1.5073	-1.3210	-18.3253	-22.4589	641.8811	1304.1243
Std	$7.19E+30$	$3.43E+39$	0.0481	0.0637	0.6863	0.8559	25.5299	31.8507
$MPEDE$								
Moy	$-7.73E+26$	$-6.45E+33$	-1.1478	-1.1488	-15.7120	-18.7170	751.2600	1415.8000
Std	$2.32E+27$	$2.83E+34$	0.0934	0.1307	0.8618	1.1669	17.3540	24.8860
MEA								
Moy	**-6.2E+34**	**-2.5E+46**	-3.4750	-3.4750	-17.8565	-19.5029	**2.7242**	8.3015
Std	**1.0E+33**	**1.9E+45**	0.2483	0.24875	1.9161	2.5603	**14.7112**	36.6508
$EAOG$								
Moy	$-4.5E{+}33$	$-1.2E{+}44$	-1.2451	-0.5902	-22.7001	-26.5300	4.8971	**4.8902**
Std	$6.9E{+}33$	$2.3.E{+}44$	1.3707	1.0172	5.0026	7.5531	4.7919	**5.0222**

Tableau 4.33 – Résultats expérimentaux obtenus pour les fonctions $F_{31} - F_{34}$

Dimension	T_0	T_1	$T2_{moy}$	$(T2_{moy}$-$T_1)/T_0$
10	$2,34$	$15,95$	$16,04$	$0,04$
30	$2,34$	$29,72$	$28,38$	$-0,57$
50	$2,34$	$41,64$	$41,23$	$-0,18$
100	$2,34$	$74,3$	$73,48$	$-0,35$

Tableau 4.34 – Résultats obtenus pour les temps T_0, T_1, $T2_{moy}$, et $(T2_{moy}$- $T_1)/$ T_0 concernant un algorithme évolutionnaire classique.

Dimension	T_0	T_1	$T2_{moy}$	$(T2_{moy}$- $T_1)/$ T_0
10	$0,1$	$448,506$	$474,7544$	$-97,516$
30	$0,1$	$1120,45$	$1105,278$	$-151,72$
50	$0,1$	$2032,6$	$2113,82$	$812,1$
100	$0,1$	$6105,23$	$6086,158$	$-190,72$

Tableau 4.35 – Résultats obtenus pour les temps T_0, T_1, $T2_moy$, et $(T2_moy$- $T_1)/$ T_0 concernant le MEA.

Dimension	T_0	T_1	$T2_moy$	$(T2_{moy}$- $T_1)/$ T_0
10	$28,09$	$9,04$	$30,05$	$0,75$
30	$25,05$	$64,3$	$55,12$	$-0,37$
50	$26,36$	$83,74$	$99,64$	$0,6$
100	$30,24$	$172,24$	$164,9$	$-0,24$

Tableau 4.36 – Résultats obtenus pour les temps T_0, T_1, $T2_moy$, et $(T2_moy$- $T_1)/$ T_0 concernant le EAOG.