ÉCOLE DE TECHNOLOGIE SUPÉRIEURE
UNIVERSITÉ DU QUÉBEC

MÉMOIRE PRÉSENTÉ À
L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

COMME EXIGENCE PARTIELLE
À L'OBTENTION DE LA
MAÎTRISE EN GÉNIE ÉLECTRIQUE
M.Ing.

PAR
BARRETTE, Mathieu

MÉTHODE DE COMPARAISON STATISTIQUE DES PERFORMANCES D'ALGORITHMES ÉVOLUTIONNAIRES

MONTRÉAL, LE 25 SEPTEMBRE 2008

CE MÉMOIRE A ÉTÉ ÉVALUÉ

PAR UN JURY COMPOSÉ DE :

M. Tony Wong, directeur de mémoire
Département de génie de la production automatisée à l'École de technologie supérieure

M. Bruno De Kelper, codirecteur de mémoire
Département de génie électrique à l'École de technologie supérieure

M. Thien-My Dao, président du jury
Département de génie mécanique à l'École de technologie supérieure

M. Marc Paquet, membre du jury
Département de génie de la production automatisée à l'École de technologie supérieure

IL A FAIT L'OBJET D'UNE SOUTENANCE DEVANT JURY ET PUBLIC

LE 7 AOÛT 2008

À L'ÉCOLE DE TECHNOLOGIE SUPÉRIEURE

MÉTHODE DE COMPARAISON STATISTIQUE DES PERFORMANCES D'ALGORITHMES ÉVOLUTIONNAIRES

BARRETTE, Mathieu

RÉSUMÉ

Il existe un nombre impressionnant de métaheuristiques pour l'optimisation globale. Les performances de ces algorithmes varient énormément, et comme le stipule le *No Free Lunch Theorem*, elles sont intimement liées à la structure du problème à optimiser et au paramétrage de l'algorithme. Le choix d'une métaheuristique et de son paramétrage pour un problème donné est une tâche complexe qui nécessite une bonne connaissance des algorithmes et du problème à optimiser. Une compréhension approfondie des comportements des métaheuristiques, de leur performance et de leur spécificité (le *No Free Lunch Theorem*) constitue une première étape vers une meilleure utilisation de celles-ci. Toutefois, étant des processus stochastiques itératifs, l'évaluation des performances de ces algorithmes requiert l'utilisation d'outils statistiques appropriés. Il n'existe actuellement aucune méthodologie universelle de comparaison des performances des métaheuristiques. Nous proposons donc ici une méthodologie générique statistiquement rigoureuse de comparaison relative des performances avec un risque d'erreur contrôlé. La méthodologie proposée est non paramétrique et utilise le rang des observations pour évaluer leurs différences. L'utilisation du rang permet ici à la fois de faire une étude non paramétrique mais permet aussi de consolider sous un seul indice de qualité, sans ajout de pondérations, les trois mesures de performance généralement observées, soit le nombre d'itérations moyen requis pour l'atteinte de l'optimum global, le niveau d'optimisation moyen et finalement, le taux d'atteinte de l'optimum global. Nous proposons également une étude complète pour une plage de paramétrage temporel choisi. Contrairement aux comparaisons habituelles d'un seul paramétrage temporel, cette façon de faire reflète beaucoup mieux le comportement temporel global des métaheuristiques et rend l'étude beaucoup plus complète.

Mots-clés : algorithme évolutionnaire, métaheuristique, optimisation, comparaison de performance, test d'hypothèse, comparaison multiple

EVOLUTIONARY ALGORITHMS PERFORMANCE EVALUATION USING RANK-BASED MULTIPLE COMPARISON PROCEDURE

BARRETTE, Mathieu

ABSTRACT

An impressive number of metaheuristics for global optimization have been developed since the apparition of the concept in the early 60's. As the *No Free Lunch Theorem* proved it, the performance of a metaheuristic is strongly related to its parameterization and to the structure of the problem to optimize. The choice of the optimal metaheuristic and its best parameterization for a specific problem is still a complex task and rely almost only on the intuition and the experience of the practitioner. A performance evaluation procedure of those processes can significantly help to understand the specificity and the evolving scheme of the metaheuristics. As metaheuristics are stochastic process, it must be done with the proper statistical tools. We propose a statistically rigorous methodology to compare the performance of different stochastic optimization algorithms. Our methodology is nonparametric and uses the rank of the data. The use of the rank instead of the true value of the data is statistically less powerful but it has the advantage to unify the three major performance criteria under a unique ranking number, without the use of arbitrary artifacts such as weighting coefficients. We also extend the analysis to a complete temporal range which gives an accurate representation of the performance on a complete range of temporal variations.

Keywords: metaheuristic, evolutionary algorithm, optimization, performance comparison, hypothesis testing, multiple comparisons

TABLE DES MATIÈRES

Page

LISTE DES TABLEAUX

Page

LISTE DES FIGURES

LISTE DES ABRÉVIATIONS, SIGLES ET ACRONYMES

AE	Algorithme Évolutionnaire
DE	Évolution différentielle (Differential Evolution)
ES	Stratégie d'évolution (Evolution Strategy)
FWER	Family Wise Error Rate
HM	Mémoire d'harmonie (Harmony Memory)
HMS	Dimension de la mémoire d'harmonie (Harmony Memory Size)
HS	Fouille par harmonie (Harmony Search)
HSD	Tukey's honestly significant difference criterion
LP	Programmation linéaire (Linear Programming)
MCP	Méthode de comparaison multiple (Multiple Comparison Procedure)
NFL	No Free Lunch Theorem
NLP	Programmation non linéaire (Nonlinear Programming)
PSO	Optimisation par essaim de particules (Particle Swarm Optimisation)
SHCLVND	Descente de gradient avec apprentissage par distribution gaussienne (Stochastic Hill Climbing with Learning by Vector of Normals Distributions)
TSP	Problème du voyageur de commerce (Traveling Salesman Problem)

INTRODUCTION

L'optimisation, au sens large, revêt aujourd'hui plus que jamais une importance capitale dans toutes les sphères de notre société. Les scientifiques, ingénieurs et décideurs sont confrontés quotidiennement à des problèmes où l'objectif primaire est de trouver une solution optimale selon un jeu de paramètres donné. Par contre, la recherche de l'optimalité n'est pas triviale et la complexité toujours grandissante des problèmes limite sensiblement les capacités et l'applicabilité des méthodes d'optimisation classiques garantissant l'optimum global. Dans de nombreux cas, l'utilisation des méthodes classiques implique une modélisation très simplifiée du problème à optimiser, limitant alors énormément leur efficacité. Pire encore, elles sont parfois inutilisables puisque les prérequis nécessaires à leur fonctionnement ne peuvent tout simplement pas être rencontrés. Toutefois, vers la fin des années 50, des travaux sur l'utilisation de méthodes stochastiques pour l'optimisation ouvraient la voie au développement d'une nouvelle branche de méthodes alternatives qui, sans garantir l'optimum global, permettent d'obtenir des résultats très acceptables tout en étant hautement génériques. Par contre, les performances de ces méthodes de dernier recours dépendent à la fois du paramétrage de l'algorithme et de la structure du problème, si bien qu'il n'existe aucun algorithme d'optimisation stochastique universellement plus performant pour tous les problèmes. Le choix d'une heuristique et de son paramétrage est donc un facteur extrêmement déterminant de la qualité des résultats. Au-delà du développement de nouvelles heuristiques d'optimisation, la caractérisation des performances et de la spécificité de ces algorithmes est un facteur aussi déterminant pour l'amélioration du processus d'optimisation globale.

Devant l'absence de procédure et de protocole d'évaluation de performance statistiquement rigoureux, il est difficile de comparer les performances d'algorithmes issus de différentes recherches. L'objectif de ce travail est de définir un protocole, statistiquement rigoureux et significatif, de comparaison des performances d'un groupe d'heuristiques d'optimisation, qui reflète au mieux les performances réelles des algorithmes en tant que systèmes stochastiques.

Le premier chapitre permettra de définir formellement le concept d'optimisation global et d'exposer les limites des méthodes classiques d'optimisation et les fondements des métaheuristiques. Le principe de la spécificité des métaheuristiques (*No free lunch theorem* Wolpert & Macready (1997)) sera ensuite exposé et expliqué en détails. Finalement, un modèle statistique des méthodes d'optimisation stochastiques sera énoncé.

Le deuxième chapitre porte exclusivement sur le processus de comparaison des performances d'algorithmes stochastiques. D'abord, une mise en contexte permettra de bien poser le cadre d'une étude de la performance des algorithmes puis, les principales faiblesses des méthodes de comparaisons actuelles seront soulevées. Pour terminer, le développement du protocole de comparaison suggéré sera détaillé.

Le troisième chapitre permettra de définir un cadre expérimental pour l'évaluation de cinq algorithmes évolutionnaires à l'aide du protocole de comparaison proposé. En premier lieu, un survol des notions fondamentales des algorithmes évolutionnaires, une sous-classe des métaheuristiques sera effectué. Puis, la structure des cinq algorithmes évolutionnaires étudiés sera explicitée. Ensuite, les différentes caractéristiques structurelles d'un problème d'optimisation et la notion de classe des problèmes seront abordées permettant ainsi de définir un banc de test représentatif constitué de problèmes synthétiques.

Le quatrième chapitre servira à définir l'ensemble du banc de test utilisé. Premièrement, les différentes caractéristiques structurelles d'un problème d'optimisation et la notion de classe des problèmes seront abordées permettant ainsi de définir un banc de test représentatif constitué de problèmes synthétiques. Ensuite, chacun des problèmes utilisés sera défini explicitement.

Le cinquième chapitre sera composé d'une analyse détaillée des résultats obtenus lors de l'application du cadre expérimental. L'analyse portera essentiellement sur les principales caractéristiques de la méthodologie proposée et sur la manière d'interpréter les résultats issus de cette méthode. Une attention particulière sera également portée sur la comparaison entre

les résultats obtenus à l'aide de notre méthodologie proposée versus les méthodologies d'analyses habituelles.

CHAPITRE 1

OPTIMISATION ET MÉTAHEURISTIQUES

1.1 Optimisation globale monocritère

L'optimisation globale est un terme très large qui inclut une grande variété de problèmes pouvant être séparés en plusieurs classes distinctes. Un problème d'optimisation peut être soit combinatoire ou continu ou même les deux à la fois. Il existe des problèmes monocritères et des problèmes multicritères et il est également fréquent qu'aux problèmes d'optimisation s'ajoutent certaines contraintes. Il existe aussi une classe distincte de problèmes de satisfaction de contraintes où l'objectif est tout simplement de trouver une solution qui répond à un certain nombre de contraintes très restrictives. Dans le présent document, nous nous attarderons exclusivement aux problèmes continus monocritères sans contrainte (mis à part un ensemble de bornes limitant la dimension du domaine de fouille). Formellement, une optimisation monocritères est le processus de recherche du minimum global (minimisation) ou du maximum global (maximisation) d'une fonction d'objectif f à l'intérieur d'un domaine O de fouille et peut être défini de la façon suivante (dans le cas d'une minimisation) selon Coello, Van Veldhuizen & Lamont (2005) :

Définition 1.1 *Soit une fonction* $f:\Omega \subseteq O = \mathbb{R}^d \to \mathbb{R}$, $\Omega \neq \varnothing$, *pour* $\vec{x} \in \Omega$, *la valeur* $f^* \triangleq f\left(\vec{x}^*\right) > -\infty$ *est un minimum global si et seulement si*

$$\forall \vec{x} \in \Omega : f\left(\vec{x}^*\right) \leq f\left(\vec{x}\right) \tag{1.1}$$

Ainsi, $\vec{x}^*$ *correspond au minimum global de la fonction d'objectif* f *et l'ensemble* Ω *est le domaine de fouille réalisable*$(\Omega \subseteq O)$. *Le problème de déterminer la solution minimum globale est un* ***problème d'optimisation global***. □

Définition 1.2 *Les paramètres à ajuster d'un problème d'optimisation sont les variables de décision notés* $L_j \leq x_j \leq U_j$ *pour* $j = 1,2,...,d$, *où* L_j *et* U_j *sont respectivement les bornes inférieures et supérieures des variables de décision. Le vecteur* $\vec{x}$ *composé des* d *variables de décision d'un problème représenté par l'équation(1.2) est appelé* ***solution***.

$$\vec{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix} \tag{1.2}$$

□

Mentionnons également que dans le cas présent, il a été considéré qu'optimiser consiste à minimiser. Toutefois, cela n'induit pas de perte de généralité puisque minimiser $f(\vec{x})$ équivaut à maximiser $-f(\vec{x})$ et $\min\{f(\vec{x})\} = -\max\{-f(\vec{x})\}$. Aussi, pour la totalité de ce mémoire, il sera désormais considéré qu'optimiser consiste à minimiser.

1.2 Limite des méthodes classiques

Il existe un certain nombre de méthodes d'optimisation classiques dites exactes. Une méthode est exacte lorsqu'elle garantit l'obtention de l'optimum global. Toutefois, le champ d'application de ces méthodes est limité à certains types de problèmes. Par exemple, la programmation linéaire (LP) n'est applicable que pour l'optimisation de problèmes continus dont la fonction de coût et les contraintes sont de nature linéaire. L'application d'un modèle linéaire simplifié pour la résolution d'un problème non-linéaire donne bien entendu des résultats optimaux pour le modèle mais reflète souvent bien mal le problème réel. Pour que la programmation non-linéaire (NLP) puisse garantir l'atteinte de l'optimum, les fonctions de coûts doivent être différentiables. Les performances de la très connue méthode de séparation et évaluation (branch & bound) dépendent entièrement de la qualité de l'algorithme de séparation utilisé. Toutefois, en pratique, un tel algorithme est difficile à trouver et intimement lié au problème à optimiser. Il est donc possible que l'algorithme ne puisse

garantir l'optimum global dans un délai raisonnable. Ce sont là trois exemples de méthodes d'optimisation classiques reconnues qui ne peuvent être utilisées que pour certains types de problèmes. C'est malheureusement la problématique des algorithmes classiques garantissant l'optimum global, ils ne sont pas applicables à tous les problèmes et requièrent certains prérequis souvent impossibles à satisfaire. Aussi, ils sont difficilement adaptables d'un problème à un autre, de sorte que leur développement peut être long et laborieux. Il existe donc une large famille de problèmes difficiles à optimiser via des méthodes exactes.

1.3 Concepts et application des métaheuristiques

À l'opposé des algorithmes d'optimisation exacte classiques, il existe d'autres méthodes, les métaheuristiques, qui sans garantir l'optimum absolu, peuvent fournir d'excellentes solutions. En plus d'être beaucoup plus génériques et facilement applicables, ces méthodes possèdent l'immense avantage d'être beaucoup moins contraignantes de sorte qu'elles sont pratiquement toujours applicables. Ainsi, dans de nombreux cas, les métaheuristiques deviennent la seule option performante envisageable. Ce sont en quelque sorte des méthodes de dernier recours. Contrairement aux méthodes mathématiques nécessitant de modéliser le problème à optimiser, ces méthodes sont basées sur la simulation. Voyons maintenant en détail les fondements directeurs des métaheuristiques à l'aide d'une étude de cas.

1.3.1 Étude de cas

Considérons un problème d'optimisation monocritère quelconque pour lequel nous ne possédons aucun indice concernant l'optimum global ou toute autre propriété mathématique permettant de le résoudre analytiquement. Supposons également que la fonction d'objectif soit non différentiable, non convexe et qu'elle possède un nombre élevé de minimums locaux. De plus, considérons que le domaine de fouille Ω, borné ou non, soit tout simplement trop grand pour que nous puissions en énumérer tous les éléments et que nous ne connaissions strictement rien de la structure du problème. Cela signifie qu'à tout moment, seul un sous-ensemble $A \subset \Omega$ ne pourra être connu. L'ensemble A est le résultat d'un échantillonnage de Ω. La solution optimale sur Ω étant $\vec{x}^*$, alors notons par $\vec{x}^{'}$ la solution

optimale de A où $f(\vec{x}^*) \leq f(\vec{x})$. Devant un problème ayant ces caractéristiques, deux constats s'imposent :

- Il est impossible de s'assurer que $\vec{x}^*$ appartienne à A.
- Dans le cas où $\vec{x}^*$ appartient à A, il est impossible de savoir si $\vec{x}^* \triangleq \vec{x}$.

De ces constatations, le vrai problème consiste maintenant à créer un algorithme d'échantillonnage permettant que $\vec{x}$ soit optimal, et qu'idéalement $\vec{x}^* \triangleq \vec{x}$ (sans que nous puissions malgré tout le prouver).

Initialement, l'ensemble A étant vide, il n'existe aucune raison directe de privilégier l'évaluation d'une solution plutôt qu'une autre. Considérons maintenant qu'un processus d'échantillonnage quelconque ait été itéré plusieurs fois de sorte que l'ensemble A contienne maintenant plusieurs éléments. Un échantillonnage purement aléatoire sur Ω est en soi un algorithme d'optimisation puisque le simple fait d'ajouter des éléments dans A augmente les probabilités d'obtenir une solution de meilleure qualité. Par contre, chaque élément de A représente une information supplémentaire sur la structure de l'espace des solutions. Ces informations partielles peuvent être exploitées pour améliorer le processus d'échantillonnage en le guidant vers des zones qui pourraient contenir des solutions de meilleure qualité. C'est précisément la manière de guider la fouille qui définit les heuristiques d'optimisation.

En pratique, les connaissances acquises de la structure de l'espace des solutions sont habituellement très sommaires et le principe directeur pour l'utilisation des connaissances est très simpliste et consiste à faire converger les recherches autour des solutions de A qui sont de meilleure qualité. Pour cela, on part d'une solution et on évalue son voisinage rapproché tant qu'une meilleure solution n'a pas été trouvée. Dès qu'une nouvelle solution est meilleure, on évalue le voisinage de cette nouvelle solution et ainsi de suite jusqu'à ce qu'il n'y ait plus d'amélioration possible. Toutefois, l'application rigide de cette règle unique donne des algorithmes d'optimisation locale basés sur la descente qui ne sont fonctionnels que pour les problèmes unimodals puisque l'algorithme converge rapidement vers un

optimum local sans pouvoir en ressortir. En fait, l'algorithme énoncé ici est un algorithme classique fonctionnel uniquement lorsque le domaine du problème est entièrement convexe et ne possède qu'un seul optimum. Il faut donc relaxer cette règle et la rendre plus souple pour que les algorithmes puissent s'extirper des attracteurs locaux et poursuivre l'optimisation vers d'autres zones potentiellement meilleures.

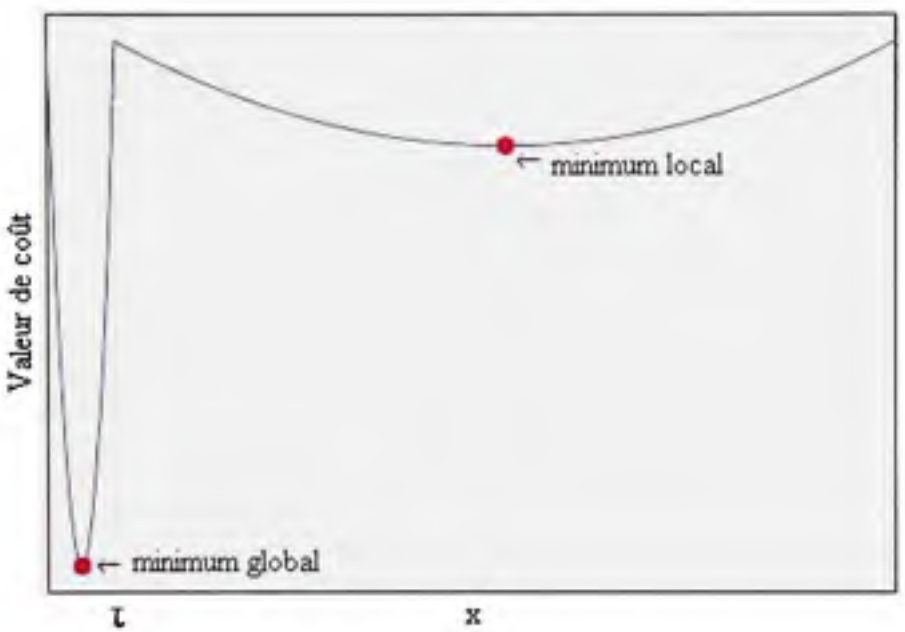

Figure 1.1 ***Fonction d'objectif possédant un minimum local et un minimum global.***

La figure 1.1 représente clairement l'importance de bien combiner l'exploitation des solutions déjà échantillonnées et l'exploitation de nouvelles zones. En effet, les informations issues d'échantillonnages antérieurs sont limitées et souvent trompeuses. La zone d'attraction vers le minimum local situé à droite de τ représente 96% du domaine de la fonction tandis que la zone d'attraction vers l'optimum global ne couvre que les 4% restant. Il est donc beaucoup plus probable qu'initialement, les premiers échantillons se situent dans la zone d'attraction locale. L'exploitation seule de ces solutions ne fournirait aucun indice permettant d'inférer sur la structure située à gauche de la limite τ et ne pourrait que faire converger les éléments vers l'optimum local. Il est primordial d'exploiter les solutions de manière à atteindre le minimum de la zone de convergence dans laquelle le processus se situe puisque

chaque optimum local est possiblement l'optimum global. Toutefois, un processus d'exploration plus global est nécessaire pour permettre d'atteindre plusieurs zones de convergence.

1.3.2 Approches métaheuristiques

Les métaheuristiques sont des processus qui tentent d'extrapoler la structure des problèmes en utilisant un haut niveau d'abstraction fortement inspiré, la plupart du temps, par des processus naturels divers. Par exemple, les algorithmes génétiques et les stratégies d'évolution sont fortement inspirés de la biologie et des principes de l'évolution darwinienne. Le recuit simulé est fortement inspiré d'un processus métallurgique. Certains algorithmes comme les colonies de fourmis et l'optimisation par essaims particulaires s'inspirent des interactions entre les individus de groupes sociaux et la recherche taboue utilise des mécanismes propres à la mémoire humaine. Le caractère stochastique de tous ces algorithmes permet d'éviter d'utiliser des règles strictes devant l'absence de certitude et ainsi de faire face au nombre démesuré d'élément de Ω. Ces méthodes sont très proches de l'intelligence artificielle. Précisément, une métaheuristiques est un algorithme d'optimisation itératif non-déterministe utilisant des processus aléatoires.

1.3.3 Limites et spécificités des algorithmes heuristiques

Les métaheuristiques peuvent être extrêmement performantes mais peuvent également donner de très mauvais résultats. La performance d'une métaheuristique dépend de plusieurs facteurs. Premièrement, une métaheuristique contient toujours un ensemble de paramètres permettant de modifier et moduler son comportement. Malheureusement, il existe très peu de règles universelles permettant de connaître le paramétrage idéal et généralement, seules l'expertise et l'expérience de l'utilisateur permettront de faire un choix. Au delà du paramétrage, il existe également un lien intime entre la performance d'un algorithme et le problème à optimiser. Ce lien est d'autant plus fort qu'il varie énormément d'un algorithme à un autre. C'est la spécificité des algorithmes qui cause ces variations.

En fait, le concept de complexité d'un problème est difficile à définir. Comme expliqué par Macready & Wolpert (1995), la complexité d'un problème est relative et inhérente à l'algorithme pour lequel la complexité est évaluée. Un algorithme d'optimisation basé sur une règle heuristique utilise les échantillons antérieurs pour évoluer selon un schéma donné défini par la structure même de l'algorithme. La complexité dépend donc directement de la "direction" que prend l'algorithme selon les informations recueillies. Un problème considéré "complexe", relativement à un algorithme donné, en sera un qui piège le schéma d'évolution de l'algorithme en lui faisant prendre des mauvaises directions ou en limitant l'efficacité de son "analyse" encapsulée à même sa structure. Vu par un autre algorithme ayant un schéma d'évolution différent, le problème pourrait fort bien être simple à optimiser. Cette conception de la complexité amène à dire que l'efficacité des algorithmes dépend de la structure du problème auquel il fait face, soit la classe du problème.

Un peu plus tard, Wolpert & Macready (1997) énoncèrent un théorème d'impossibilité nommé *No Free Lunch Theorem* (NFL). Ce théorème stipule que sur l'ensemble de tous les problèmes d'optimisation possibles, tous les algorithmes d'optimisation offrent des performances identiques. La meilleure performance d'un algorithme a par rapport à un algorithme b pour un problème donné signifie simplement que l'algorithme a est mieux adapté à ce problème et qu'il est alors nécessairement moins performant que b sur l'ensemble des autres problèmes d'optimisations possibles. Ce théorème permet fondamentalement de démontrer la spécificité des algorithmes et s'applique même à la pure fouille aléatoire. Le NFL stipule donc que devant l'absence de connaissance a priori nous ne pouvons pas déterminer quel algorithme d'optimisation performera le mieux. L'idée de se fier aux performances d'un algorithme sur un problème donné pour inférer sur ses possibles performances sur un autre problème est plutôt risquée. Cela amène donc un autre problème qui est celui de la définition d'une classe de problèmes. Comment définit-on une classe de problèmes? Qu'elles sont les caractéristiques qui définissent l'appartenance d'un problème à une classe? Considérons par exemple le problème du voyageur de commerce (Traveling Salesman Problem TSP). Un praticien ayant développé un algorithme permettant d'optimiser

efficacement ce problème a en réalité développé un algorithme dont la spécificité est fortement orientée vers ce type de problème.

Toutefois, selon Weinberg & Talbi (2004), le NFL est trop général et omet certaines considérations sur les problèmes susceptibles de devoir être optimisés. Le fait soulevé par ces derniers est que ce ne sont pas tous les problèmes qui doivent être optimisés mais bien certaines classes de problèmes seulement. Aussi, ils démontrent via les problèmes de coloration de k-graphes, qu'il peut exister une certaine structure commune aux problèmes d'une même classe et, dans la mesure où l'on reste à l'intérieur d'une certaine classe, le NFL ne tient plus. Le même phénomène est présent pour le problème TSP précédemment énoncé. Malgré le fait que le praticien n'ait clairement pas évalué son algorithme sur la totalité des problèmes TSP à N villes, l'algorithme risque fort bien d'être efficace pour les problèmes TSP en général sans que l'on ne sache exactement ce qui caractérise cette classe de problèmes.

En somme, il est difficilement imaginable qu'un seul algorithme puisse être le plus efficace pour l'ensemble des problèmes d'optimisation réels auxquels nous pouvons être confrontés. Cette notion de classe démontre la nécessité de comparer les performances des métaheuristiques sur des bancs de tests couvrant plusieurs classes de problèmes permettant ensuite de les classifier.

1.4 Modélisation statistique des métaheuristiques

Une métaheuristique est avant tout un processus stochastique itératif où une itération est l'ajout d'un élément de Ω à l'ensemble A des solutions échantillonnées. À chaque itération, l'algorithme fournit le vecteur solution de A dont la fonction de coût est la meilleure et qui est égale ou inférieure (dans le cas d'une minimisation) à celle de l'itération précédente. Dans notre cas, la nature même du vecteur solution n'est d'aucun intérêt, nous nous intéressons plutôt à la valeur de coût ϕ des solutions, où $\phi \triangleq f(\vec{x})$. Nous savons également qu'une métaheuristique, étant un processus stochastique, est non déterministe. Cela signifie

que deux exécutions successives du même algorithme ayant le même paramétrage, appliquées à un même problème risquent fortement de donner deux solutions différentes. Définissons exhaustivement quelques termes statistiques qui seront utilisés pour la modélisation.

Définition 1.3 *Une **variable aléatoire**, notée X est une fonction dont la probabilité de réalisation de chacune des valeurs de l'espace échantillonnal S est définie et suit une loi de probabilité quelconque.* □

Définition 1.4 *L'ensemble S (discret ou continu), nommé l'**espace échantillonnal**, constitue l'ensemble des valeurs que peut prendre une variable aléatoire X.* □

Définition 1.5 *Une **loi de probabilité** ou **distribution de probabilité** est une fonction qui détermine la probabilité de chaque valeur de S lors de la réalisation d'une variable aléatoire.* □

Définition 1.6 *Un **individu** ou une **observation** est une réalisation d'une variable aléatoire.* □

Définition 1.7 *Un **échantillon** est un ensemble de réalisations d'une variable aléatoire constituée de plusieurs observations.* □

Définition 1.8 *Un **processus stochastique** $\mathbf{X} = \{X(t), t \in T\}$ est une suite de variables aléatoires. Ainsi, à chaque index $t \in T$ est associé une variable aléatoire $X(t)$. Généralement, t est un indicateur temporel, toutefois, t peut également être multidimensionnel. L'ensemble T peut être discret (processus stochastique discret) ou continu (processus stochastique continu). Lorsque T est discret, nous utilisons plutôt la notation $n \in N$ où $\mathbf{X} = \{X(n), n \in N\}$. Nous nommons **trajectoire**, la réalisation d'un processus stochastique.* □

Définition 1.9 *Soit une fonction* $f : \mathbb{R} \to \mathbb{R}$ *et un intervalle* I. *La fonction* f *est dite monotone décroissante sur l'intervalle* I *si pour tout couple d'éléments* (x_a, x_b) *de* I, *si* $x_a \leq x_b$, *alors* $f(x_a) \leq f(x_b)$. □

Une métaheuristique est un processus stochastique $\Phi = \{\phi(n), n = \{1,2,....,\eta\}\}$, où η est le nombre total d'itérations effectuées par l'algorithme. Chaque instance de ce processus est une trajectoire particulière issue des η distributions de probabilité formant le processus d'optimisation. De plus, toute réalisation d'un processus d'optimisation itératif est implicitement une trajectoire monotone décroissante. Dû à cette caractéristique, le domaine des états possibles S est dépendant de n de sorte que $S[n+1]$ appartient à l'intervalle $[f^*; \phi(n)]$. Il existe donc une forte corrélation entre la $n^{ième}$ itération d'un processus et les $n-1$ distributions précédentes. Cette corrélation rend le processus non stationnaire puisqu'un processus stochastique possède la stationnarité si la probabilité de distribution de $X(n+u) - X(n)$ est la même pour tout n et $u > 0$.

Aussi, lors d'un processus d'optimisation, où l'optimum global est inconnu, le nombre d'itérations η de la trajectoire est déterminé par le praticien à l'aide d'un critère d'arrêt. Le mécanisme d'arrêt peut être un nombre d'itérations à effectuer, un temps de calcul prédéterminé ou encore, un critère dynamique basé sur la trajectoire. Dans un tel contexte, la solution optimale correspond à celle ayant le plus haut niveau d'optimisation (plus petite valeur de coût) soit l'état $\phi(\eta)$ car pour un nombre d'itérations donné, la dernière solution est toujours celle dont la valeur de coût ϕ est la meilleure. D'autres solutions peuvent avoir une valeur de coût égale à celle de la dernière itération mais techniquement, il n'y a aucun avantage à favoriser une autre solution que $\phi(\eta)$. Pour le praticien, le processus d'optimisation fournit une seule solution optimale et les $\eta - 1$ solutions issues des itérations antérieures sont toutes des solutions sous optimales sans intérêt.

CHAPITRE 2

ANALYSE DE LA PERFORMANCE DES MÉTAHEURISTIQUES

2.1 Contexte de l'analyse

La comparaison des performances de processus stochastiques d'optimisation nécessite l'utilisation d'outils statistiques appropriés. Le présent chapitre servira à expliquer les diverses caractéristiques définissant la performance de processus stochastiques d'optimisation puis à exposer les faiblesses des méthodes de comparaison actuellement utilisées. Finalement, une nouvelle méthode statistiquement plus rigoureuse et plus représentative de la performance des processus d'optimisation sera proposée.

Pour un praticien, le nombre d'itérations à effectuer est un paramètre de premier plan pouvant avoir un impact majeur sur le niveau d'optimisation de la solution finale. L'impact de ce critère dépend à la fois de l'algorithme d'optimisation (et de son paramétrage initial) et du problème à optimiser. Dans un contexte d'évaluation de la performance des algorithmes stochastiques, chacune des η solutions d'un processus stochastique doivent être considérées puisqu'elles sont toutes des solutions optimales pour un paramétrage donné correspondant aux limites temporelles d'un praticien. Avant même de vouloir répondre à la question : "*Quel algorithme est susceptible de me donner la meilleure solution ?*", un praticien doit se demander : "*Combien de temps puis-je allouer à l'optimisation ?*". Par exemple, une solution obtenue après $n > 0$ itérations, peut fort bien être sous optimale pour un praticien ayant itéré le processus $2n$ fois mais peut être sans contredit la meilleure solution pour un praticien n'ayant pu itéré le processus que n fois. Tout comme il est possible qu'un processus a soit plus performant qu'un processus b pour $2n$ itérations et que le processus b soit plus performant que le processus a pour n itérations.

Le temps alloué à un processus d'optimisation est la première contrainte d'une optimisation et il est évident que la comparaison des métaheuristiques doit se faire pour un même

paramétrage temporel correspondant au même contexte. Il serait bien injuste de comparer les résultats de deux processus où l'un aurait eu droit à deux fois plus de temps de calcul que l'autre. Idéalement, le protocole de comparaison devrait pouvoir évaluer les performances sur une fenêtre temporelle infinie avec une infinité de comparaisons, soit une par itération. Cela est toutefois impossible. Voyons quelles sont les contraintes d'une étude de performance des métaheuristiques.

2.1.1 Contraintes et particularités

2.1.1.1 Limite temporelle

Théoriquement, pour étudier les performances des métaheuristiques, nous devrions laisser itérer les algorithmes tant que leur état interne possède une probabilité d'amélioration non nulle. Toutefois, dans la majorité des cas, cet état est impossible à identifier, n'arrivera jamais où prendra beaucoup trop de temps à se produire. Nous sommes donc obligés de limiter le nombre d'itérations à $N_{\max}$ évaluations. Notre pouvoir d'analyse est limité à cette fenêtre temporelle en dehors de laquelle nous ne connaissons rien des trajectoires (sauf celles ayant déjà atteint l'optimum global). Il est évident que plus la valeur de la limite $N_{\max}$ est petite, moins l'analyse est étoffée.

2.1.1.2 Zone optimale

Dans un contexte d'évaluation de performance d'algorithmes d'optimisation stochastiques, il est préférable d'utiliser des fonctions de coûts dont l'optimum global est connu. Cela permet de savoir, lorsque cela se produit, si l'instance a été optimale ou si elle a simplement été emprisonnée par un attracteur local. Une instance ayant atteint l'optimum après n itérations donnera nécessairement le même résultat pour toutes les itérations subséquentes. Techniquement, cela nous permet donc d'interrompre un processus ayant atteint cet état sans restreindre la portée de l'analyse. Aussi, toutes les instances ayant atteint l'optimum global ne sont pas automatiquement équivalentes. Outre l'objectif ultime d'une optimisation, qui

consiste à atteindre l'optimum global, le temps nécessaire à l'atteinte de cette valeur est aussi très important. Dans le cadre d'une étude de performance, il est essentiel que pour un paramétrage temporel donné, deux solutions ayant atteint l'optimum global dans des temps différents soient différentiables.

D'un point de vue purement technique, pour assigner un niveau de qualité aux solutions, lorsqu'une instance a atteint l'optimum global, nous assignons à η de cette instance la valeur du nombre d'itérations nécessaires à son atteinte. Aussi, l'optimum est habituellement un point ponctuel, mais les systèmes informatiques ont une granularité finie qui ne permet pas d'atteindre la précision infinie d'un point ponctuel. Aussi, l'atteinte d'une précision infinie est dans notre cas d'aucun intérêt. Nous limitons donc arbitrairement la précision d'une optimisation en définissant une zone à atteindre englobant l'optimum. Les dimensions de cette zone sont définies à l'aide d'un seuil ε. L'optimisation est considérée optimale et le processus est immédiatement interrompu dès lors que $\phi(n) \leq (f^* + \varepsilon)$.

2.1.1.3 Limite de mémoire

Une bonne étude devra être faite sur une fenêtre temporelle dont la dimension dépend de l'ensemble des trajectoires des processus comparés. Souvent, le nombre d'itérations est très élevé, de l'ordre du million. La mémorisation et la comparaison de chacune des instances représentent un temps de calcul immense et requièrent énormément de mémoire. Pour limiter les besoins en mémoire, nous échantillonnons les itérations pour lesquelles une étude de la performance sera faite. La meilleure façon de faire est d'effectuer une évaluation à toutes les δ itérations. Ici δ est défini en fonction de la dimension de la fenêtre temporelle et des ressources de l'expérimentateur.

2.2 Comparaison des performances : état de l'art

Lorsqu'un chercheur propose une nouvelle méthode d'optimisation heuristique ou encore une variante d'une méthode déjà existante, il se doit d'exposer les performances de sa

proposition. Toutefois, la façon de mesurer les performances des algorithmes diffère énormément d'une étude à l'autre et cela rend les comparaisons inter études ardues. En réalité, aucune méthodologie de comparaison universelle n'existe actuellement et il n'en revient qu'au chercheur d'évaluer ce qui l'intéresse de la manière qu'il le désire. Aussi, les méthodes de comparaisons sont généralement inappropriées et peu représentatives de la performance réelle des algorithmes en tant que système stochastique. Voyons en premier lieu la définition de deux mesures régulièrement utilisées, puis, différentes méthodes d'analyse retrouvées.

Définition 2.1 *Soit un échantillon composé de* x *trajectoires d'une métaheuristique quelconque, parmi lesquelles* $y \leq x$ *trajectoires ont atteint l'optimum global. La* ***vitesse de convergence*** *de cet échantillon représente, le nombre d'itération moyen ayant été nécessaire aux* y *itérations pour atteindre l'optimum global. Les* $(x-y)$ *solutions n'ayant pas atteint l'optimum ne sont pas représentées par cette mesure.* □

Définition 2.2 *Le* ***taux de convergence*** *d'un échantillon, est une mesure représentant le taux d'atteinte (en pourcentage) de l'optimum global.* □

Par exemple, dans le cas où l'optimum global est connu, Storn & Price (1997) mesure la performance des algorithmes à l'aide de la vitesse de convergence des échantillons. Aussi, une indication permettant de savoir qu'un algorithme n'a pas atteint l'optimum à tous les coups est ajoutée, sans toutefois donner plus d'informations sur le taux d'atteinte ni sur le niveau d'adaptation de ces solutions. Les instances n'ayant pas atteint l'optimum sont alors vues comme des échecs et ne font pas partie des résultats. Le rejet systématique des observations n'ayant pas convergées peut limiter l'analyse à certains niveaux et induit même un biais dans certains cas. La connaissance de l'optimum d'une fonction d'un banc de test est une information qui donne effectivement un pouvoir supplémentaire d'analyse mais la création d'un seuil utilisé de la manière explicitée précédemment entraîne plutôt une perte de généralité. En fait, une telle utilisation élimine entièrement la notion de qualité de la solution limitant l'analyse à la vitesse de convergence et à un taux de convergence basé sur le critère

ε et sur un seul état du paramètre N_{max}. On ne compare donc plus le niveau d'optimisation des solutions. Si le seuil est atteint, on analyse la convergence, sinon, on ne l'analyse pas. Pourtant, la qualité d'une solution peut varier énormément. Prenons comme exemple, la fonction de coût de Rastrigin à 100 dimensions. Avec les bornes habituelles, la plage de valeurs que peut prendre cette fonction de coût correspond à l'intervalle $\left[0\,;\,36\times10^{6}\right]$. Pourtant, avec la méthode d'analyse classique, avec un seuil classique $\varepsilon=1\times10^{-6}$, une solution de l'ordre de 1×10^{-5} serait considérée comme un échec équivalent à une solution de l'ordre de 30×10^{6} et aucune analyse comparative de ces deux solutions ne serait faite. Il est pourtant évident que dans cette situation extrême, le premier cas représente clairement une grande faiblesse (pour le N_{max} utilisé) tandis que le second est à peine différentiable d'un succès et constitue un rejet arbitraire très discutable. Aussi, le concept de l'optimum à tout prix est totalement contradictoire à la notion de métaheuristique énoncée dans le chapitre précédent. Comme il a été mentionné, une métaheuristique est un processus qui ne garantit pas l'optimum global mais donne une approximation dans un temps raisonnable.

À l'inverse, Rudolf & Koppen (1996) donne les taux de convergences des algorithmes sans spécifier le nombre d'itérations moyen nécessaire mais seulement le nombre d'itérations maximal alloué. Ainsi, il est impossible de distinguer deux algorithmes ayant le même taux de convergence et il est impossible de connaître la vitesse d'évolution. De plus, ici aussi, nous ne connaissons strictement rien des algorithmes n'ayant pas atteint l'optimum et la méthode de comparaison ne considère qu'une seule valeur de N_{max}.

Dans un tout autre ordre d'idée, Shi & Eberhart (1999) n'évalue que la valeur de coût moyenne pour un nombre d'itérations donné (N_{max} arbitraire). Il n'est alors pas possible de distinguer des instances ayant atteint l'optimum global. Toutefois, cette procédure a l'avantage de prendre en considération toutes les instances des échantillons traités.

Dans la même optique, Geem &Tseng (2002a), Geem &Tseng (2002b) et Lee & Geem (2004) évaluent eux aussi uniquement les valeurs de coût. Toutefois, les valeurs obtenues ne

sont pas des moyennes mais semblent plutôt être soit le résultat d'une seule évaluation, ou les meilleurs cas d'échantillons de dimensions inconnues. Cette façon de comparer les résultats possède les mêmes faiblesses que la précédente en plus d'être statistiquement moins significative. De plus, le paramétrage unique de N_{max} limite encore une fois la portée de la comparaison.

2.2.1 Valeur statistique des méthodes de comparaison

Les méthodes exposées précédemment évaluent toutes les performances des algorithmes à l'aide de mesures différentes qui ne considèrent qu'une seule valeur de N_{max} et ne sont donc pas une représentation exhaustive de la performance générale de l'algorithme mais plutôt une évaluation ponctuelle pour un paramétrage donné parmi une infinité de possibilités. De plus, il existe un autre problème fondamental encore plus critique à toutes ces méthodes qui peut porter à des conclusions erronées.

Prenons comme exemple le tableau 2.1 contenant deux statistiques de cinq échantillons de 50 observations chacun, toutes issues d'une seule et unique loi de probabilité $U(0,100)$, soit une distribution uniforme sur l'intervalle $[0\ ;\ 100]$.

Tableau 2.1

Analyse de cinq échantillons issus de la même distribution

Observateur	Statistique	$\bar{x}_1$	$\bar{x}_2$	$\bar{x}_3$	$\bar{x}_4$	$\bar{x}_5$
A	$\bar{x}$	57.10	52.02	50.98	52.78	46.45
B	$\min\{x\}$	0.99	4.93	2.87	1.42	4.66

Imaginons que les données reflètent un critère de performance pour lequel plus la valeur est petite, plus la solution est de qualité. Supposons maintenant que deux observateurs extérieurs (A et B) aient chacun accès uniquement à une statistique et qu'ils soient susceptibles d'inférer des conclusions de la même manière que les chercheurs des études énoncées

précédemment. L'observateur A, qui aurait accès uniquement à la moyenne de chaque groupe, serait probablement tenté de conclure que la distribution de l'échantillon x_5 est la meilleure puisqu'elle a produit la plus petite moyenne. À l'opposé, l'observateur B ayant accès uniquement à la valeur minimale de chaque échantillon pourrait être tenté de dire que la distribution x_1 est la meilleure. Connaissant la provenance des données, nous savons pourtant que les deux observateurs ont tort puisque les cinq échantillons proviennent de la même distribution.

Cette expérience a permis de constater qu'il n'est pas acceptable de comparer de façon directe des statistiques d'échantillons de dimensions finies. Les chances qu'une statistique de deux échantillons de petites tailles issus d'une distribution de probabilité unique soient identiques sont nulles. Comparer des statistiques, tels des moyennes ou des extremums d'échantillons de petites tailles, ne permet pas d'inférer adéquatement sur les propriétés des distributions. En fait, en procédant de la sorte, les chercheurs comparent les caractéristiques des échantillons plutôt que les caractéristiques des générateurs de ces échantillons. L'objectif absolu est ici de comparer chacune des η distributions de chaque processus stochastique.

2.3 Élaboration d'une méthodologie

Nous voulons élaborer une méthodologie qui permet de mesurer la performance globale des processus d'optimisation qui serait à la fois statistiquement fiable, qui minimiserait l'utilisation de décisions subjectives et qui prendrait en compte toutes les caractéristiques de la performance d'une observation. Voyons en premier lieu, comment la valeur statistique de l'étude peut être augmentée pour l'analyse d'une seule valeur de $N_{\max}$.

2.3.1 Les tests d'hypothèses

Comme il a été soulevé, les méthodes classiques de comparaison manquent de rigueur statistique. Le but derrière une comparaison des métaheuristiques n'est pas de comparer des

échantillons mais bien de déterminer quelle distribution est la plus susceptible de fournir les meilleurs résultats. Pour répondre à cette question, nous devons utiliser un test d'hypothèse.
Un test d'hypothèse est une démarche consistant à rejeter ou à accepter une hypothèse statistique, appelée hypothèse nulle (H_0), en se basant sur un ou des échantillons. Le rejet de l'hypothèse nulle implique l'acceptation d'une hypothèse alternative (H_1). Les hypothèses H_0 et H_1 doivent être mutuellement exclusives. Dans notre cas, l'objectif escompté est de comparer les performances de différents algorithmes. Nous sommes donc intéressés par les tests statistiques d'homogénéité. Le test d'homogénéité permet d'inférer sur l'homogénéité entre plusieurs échantillons indépendants en se basant sur différentes statistiques des échantillons, telle leur moyenne. Deux échantillons sont indépendants lorsque les résultats de l'un n'influence pas les probabilités de l'autre. Lors de ce type de test, l'hypothèse nulle supposera l'homogénéité entre les échantillons tandis que H_1 supposera qu'au moins deux échantillons parmi les k échantillons comparés soient différents l'un de l'autre. Plus spécifiquement, le test permet d'inférer sur les distributions de probabilité de chaque variable aléatoire en se servant uniquement des données. Une telle analyse n'apporte pas de certitude mais permet d'obtenir une approximation à laquelle est associée une probabilité p. Évidemment, plus la taille des échantillons est importante, plus l'approximation risque d'être fiable.

Intrinsèquement, un test d'hypothèse permet d'obtenir une mesure (p) de la probabilité que l'hypothèse nulle soit vraie selon les échantillons traités. Ainsi, plus la valeur de p est petite, plus le test possède d'évidences contre la véracité de l'hypothèse nulle. Un seuil de décision α, nommé niveau de signification ou risque d'erreur, est donc associé à p de sorte que l'hypothèse nulle sera acceptée lorsque $p \geq \alpha$ et rejetée lorsque $p < \alpha$. Cela induit inévitablement deux types d'erreurs probables lors de la prise de décision.

Définition 2.1 *Une erreur de type I (faux négatif) consiste à rejeter l'hypothèse nulle lorsqu'elle est vraie.* □

Définition 2.2 *Une erreur de type II (faux positif) consiste à accepter l'hypothèse nulle lorsqu'elle est fausse.* □

Le tableau 2.2 montre les quatre situations possibles lors d'une prise de décision à l'aide d'un test d'hypothèse. La valeur β est le risque de seconde espèce, très difficilement calculable, c'est le risque d'accepter l'hypothèse nulle lorsqu'elle est fausse.

Tableau 2.2
Probabilité des erreurs de type I et II lors d'un test statistique

	État réel de l'hypothèse nulle H_0	
Décision statistique	H_0 Vraie	H_0 Fausse
Accepter H_0	Correct $(1-\alpha)$	Erreur de type I (α)
Rejeter H_0	Erreur de type II (β)	Correct $(1-\beta)$

2.3.2 Test de Kruskal-Wallis

Le test de Kruskal-Wallis est un test d'homogénéité non-paramétrique. Un test statistique est considéré non-paramétrique lorsqu'il ne fait aucune supposition sur les distributions de probabilité des échantillons traités. Il est une extension du test de Mann-Whitney-Wilcoxon applicable à plus de deux échantillons. Ce test, appliqué à k échantillons mutuellement indépendants, nous permettra de vérifier l'hypothèse (H_0) selon laquelle les échantillons proviennent de distributions de probabilités identiques. Plus précisément, le test de Kruskal-Wallis permet de déceler des différences entre les moyennes des k populations (Conover (1998)). Les hypothèses de ce test sont les suivantes :

$$H_0 : \mu_1 = \mu_2 = \ldots = \mu_k$$
$$H_1 : \mu_i \neq \mu_j \text{ où } i, j \leq k \text{ et } i \neq j$$

Le test de Kruskal-Wallis est une statistique de rang. Une statistique de rang, ou statistique d'ordre, est une statistique basée sur le rang relatif des observations des échantillons. La valeur numérique réelle de chaque observation sert exclusivement à l'attribution des rangs relatifs des observations pour la procédure statistique. La puissance statistique d'un test consiste en sa capacité à exploiter efficacement les informations des données pour faire une analyse précise sous un risque d'erreur contrôlé. L'utilisation du rang des individus plutôt que leur valeur introduit donc inévitablement une perte de puissance statistique relativement aux méthodes qui utilisent directement les données, puisque l'écart entre les valeurs ordinales des individus est riche d'information et que les tests de rang n'utilisent pas cette information. Toutefois, ce type de test convient à des situations où les prérequis des méthodes paramétriques ne sont pas satisfaits. Dans notre cas, nous avons effectivement besoin d'utiliser une méthode de comparaison non-paramétrique puisque nous ne connaissons pas les distributions des systèmes d'optimisation mais de plus, l'utilisation des rangs est dans notre cas un excellent moyen d'éviter d'utiliser des pondérations arbitraires pour quantifier la qualité des solutions comme expliqué ultérieurement à la section 2.3.4.

2.3.2.1 Assignation des rangs

Dans ce travail de recherche, nous utiliserons une fonction d'ordre permettant de classer les observations en ordre croissant de leur qualité d'optimisation, de sorte que l'observation ayant la plus petite valeur de qualité (la meilleure) reçoive le plus petit rang tandis que l'observation ayant la valeur la plus élevée (la moins bonne) reçoive le rang le plus élevé. Cette approche n'est pas la seule mais elle est généralement préférée vu sa simplicité et sa logique naturelle. Soit $Q(N_{max})$, une fonction représentant la qualité d'une observation pour un paramétrage temporel donné et $r(Q)$ une fonction permettant de lui associer son rang. Ainsi, si $Q_i \leq Q_j$, $r(Q_i) \leq r(Q_j)$. Considérons une série de M observations quelconques $Q_1, Q_2, \ldots, Q_M$ à classer. La façon la plus simple d'assigner des rangs aux M observations, consiste à leur attribuer les M premiers entiers positifs. L'équation(2.1) permet d'associer un rang R_i à l'observation Q_i

$$R_i = r(Q_i) = \sum_{j=1}^{M} E(Q_i - Q_j) \tag{2.1}$$

Où

$$E(u) = \begin{cases} 0 \text{ si } u < 0 \\ 1 \text{ si } u \geq 0 \end{cases} \tag{2.2}$$

Le fonctionnement des équations (2.1) et(2.2) est démontré à l'aide du tableau 2.3 suivant, où l'on retrouve l'assignation des rangs des sept valeurs entières désordonnées suivantes : [3, 6, 2, 11, 4, 1, 9]. Chaque case de données de la matrice représente le résultat de $Q_i - Q_j$ tel qu'utilisé à l'équation (2.1) et entre parenthèse, le résultat de l'équation (2.2). La dernière rangée, qui représente le rang des Q_i observations, est obtenue en sommant les résultats (entre parenthèse) des cellules de la colonne.

Tableau 2.3

Exemple d'assignation des rangs

		Q_i						
		3	**6**	**2**	**11**	**4**	**1**	**9**
Q_j	**3**	0 (1)	3 (1)	-1 (0)	8 (1)	1 (1)	-2 (0)	6 (1)
	6	-3 (0)	0 (1)	-4 (0)	5 (1)	-2 (0)	-5 (0)	3 (1)
	2	1 (1)	4 (1)	0 (1)	9 (1)	2 (1)	-1 (0)	7 (1)
	11	-8 (0)	-5 (0)	-9 (0)	0 (1)	-7 (0)	-10 (0)	-2 (0)
	4	-1 (0)	2 (1)	-2 (0)	7 (1)	0 (1)	-3 (0)	5 (1)
	1	2 (1)	5 (1)	1 (1)	10 (1)	3 (1)	0 (1)	8 (1)
	9	-6 (0)	-3 (0)	-7 (0)	2 (1)	-5 (0)	-8 (0)	0 (1)
	R	**3**	**5**	**2**	**7**	**4**	**1**	**6**

2.3.2.2 Gestion des égalités de rang

L'équation (2.1) suppose que les valeurs des M observations sont différentes de sorte qu'il n'y ait pas d'observations ayant le même rang. Théoriquement, si les distributions de

probabilités sont continues, la probabilité qu'une égalité entre deux observations se produise est nulle. Toutefois, en pratique, des égalités peuvent se produire. Considérons le cas où t observations auraient la même valeur, $t \leq n$. Suivant l'équation (2.1), ces t observations recevraient toutes le même rang, équivalant au plus haut rang dans le cas ou les t observations étaient différentiables. Toutefois, ce type de gestion des égalités n'est pas acceptable et introduit un biais puisque les statistiques de rangs sont basées sur le principe que les rangs sont tous différents. Une méthodologie d'ajustement des égalités est donc nécessaire pour préserver la véracité des résultats.

Il existe plusieurs techniques pour gérer les égalités entre observations (Gibbons & Chakraborti, 1992). Celle que nous utiliserons est la méthode du rang moyen, qui consiste tout simplement à attribuer un unique rang équivalant à la moyenne des rangs des t observations comme si elles étaient différentiables. L'assignation des rangs suivant la logique du rang moyen pour les observations identiques peut facilement être effectuée en modifiant l'équation (2.1) et l'équation (2.2) par l'équation (2.3) et l'équation (2.4) tel qu'exprimé par Gibbons & Chakraborti (1992).

$$R_i = r(Q_i) = 0.5 + \sum_{j=1}^{M} E(Q_i - Q_j) \tag{2.3}$$

Où

$$E(u) = \begin{cases} 0 & \text{si } u < 0 \\ 0.5 & \text{si } u = 0 \\ 1 & \text{si } u > 0 \end{cases} \tag{2.4}$$

L'exemple suivant permet de démontrer le fonctionnement des équations(2.3) et (2.4). Considérons l'échantillon constitué des 9 observations suivantes [7,7,2,4,5,11,7,9,2].

Le tableau 2.4 suivant fonctionne de la même façon que le tableau 2.3. Toutefois, une ligne est ajoutée à la suite de la matrice des différences permettant d'ajouter la valeur de 0.5 de l'équation(2.4). L'échantillon traité ici contient deux fois la valeur 2 et trois fois la valeur 7. Les deux valeurs 2 étant les plus basses de l'échantillon recevraient, si nous pouvions les différentier, respectivement les rangs un et deux. La méthode de gestion des égalités consiste à donner le rang moyen, donc dans ce cas-ci, le rang des deux observations sera de $(1+2)/2=1.5$. Suivant la même logique, les trois observations égales à sept recevraient, si nous pouvions les différentier, les rangs, cinq, six et sept. Ils reçoivent donc tous le rang $(5+6+7)/3=6$. Notons qu'en assignant le rang moyen aux observations identiques, la sommation des rangs donne le même résultat, ici 45, que celui obtenu s'il n'y avait pas eu d'égalité.

Tableau 2.4

Exemple d'assignation des rangs avec gestion des égalités

		Q_i								
		7	**7**	**2**	**4**	**5**	**11**	**7**	**9**	**2**
Q_j	**7**	0 (0.5)	0 (0.5)	-5 (0)	-3 (0)	-2 (0)	4 (1)	0 (0.5)	2 (1)	-5 (0)
	7	0 (0.5)	0 (0.5)	-5 (0)	-3 (0)	-2 (0)	4 (1)	0 (0.5)	2 (1)	-5 (0)
	2	5 (1)	5 (1)	0 (0.5)	2 (1)	3 (1)	9 (1)	5 (1)	7 (1)	0 (0.5)
	4	3 (1)	3 (1)	-2 (0)	0 (0.5)	1 (1)	7 (1)	3 (1)	5 (1)	-2 (0)
	5	2 (1)	2 (1)	-3 (0)	-1 (0)	0 (0.5)	6 (1)	2 (1)	4 (1)	-3 (0)
	11	-4 (0)	-4 (0)	-9 (0)	-7 (0)	-6 (0)	0 (0.5)	-4 (0)	-2 (0)	-9 (0)
	7	0 (0.5)	0 (0.5)	-5 (0)	-3 (0)	-2 (0)	4 (1)	0 (0.5)	2 (1)	-5 (0)
	9	-2 (0)	-2 (0)	-7 (0)	-5 (0)	-4 (0)	2 (1)	-2 (0)	0 (0.5)	-7 (0)
	2	5 (1)	5 (1)	0 (0.5)	2 (1)	3 (1)	9 (1)	5 (1)	7 (1)	0 (0.5)
		(0.5)	(0.5)	(0.5)	(0.5)	(0.5)	(0.5)	(0.5)	(0.5)	(0.5)
	R	**6**	**6**	**1.5**	**3**	**4**	**9**	**6**	**8**	**1.5**

2.3.2.3 Application du test de Kruskal-Wallis

Soit k échantillons indépendants composés chacun de m_i observations positionnées en colonnes, et chaque colonne représente un échantillon. Chacune des m_i observations $Q_{ij}, i = 1,2,...,k$, $j = 1,2,...,m_i$ des k échantillons peut être identifiée de la manière suivante :

Échantillon 1	**Échantillon 2**	...	**Échantillon** k
Q_{11}	Q_{21}	...	Q_{k1}
Q_{12}	Q_{22}	...	Q_{k2}
$\vdots$	$\vdots$	...	$\vdots$
Q_{1m_1}	Q_{2m_2}	...	Q_{km_k}

Notons le nombre total d'observations M où :

$$M = \sum_{i=1}^{k} m_i \tag{2.5}$$

La première étape du test consiste à assigner un rang aux M observations des k échantillons à l'aide de l'équation (2.3) et de l'équation (2.4). Cette transformation permet d'obtenir le rang de chacune des observations.

Échantillon 1	**Échantillon 2**	...	**Échantillon** k
R_{11}	R_{21}	...	R_{k1}
R_{12}	R_{22}	...	R_{k2}
$\vdots$	$\vdots$	...	$\vdots$
R_{1m_1}	R_{2m_2}	...	R_{km_k}

Le test de Kruskal-Wallis est défini par l'équation (2.6) suivante (Gibbons & Chakraborti, 1992) :

$$H = \frac{12}{M(M+1)} \sum_{i=1}^{k} \frac{\sum_{j=1}^{m_i} (R_{ij})^2}{m_i} - 3(M+1) \tag{2.6}$$

Lorsqu'il y a des égalités de rangs, un ajustement supplémentaire s'impose toutefois puisque même en utilisant la méthode d'ajustement des égalités, la variance des rangs lorsqu'il y a égalité diffère de la variance des rangs sans égalité. À l'aide de la définition de t à la section 2.3.2.2 et de l'équation (2.6), le test de Kruskal-Wallis devient le suivant :

$$H = \frac{H}{1 - \frac{\sum t^3 - t}{M^3 - M}} \tag{2.7}$$

En reprenant le même exemple qu'à la section 2.3.2.2, soit l'échantillon [7,7,2,4,5,11,7,9,2], nous obtiendrions :

Tableau 2.5

Ajustement pour les égalités

Observation	t	t^3
2	2	4
7	3	27
	5	31

Il est possible de déterminer la distribution exacte de H par énumération mais comme cela est plutôt laborieux et nécessite l'utilisation de tables très complexes dû aux nombreuses variantes de $M, k, \{m_1, m_2, ..., m_k\}$, la procédure propose plutôt d'approximer cette distribution à l'aide de la distribution du χ^2 à $k-1$ degrés de liberté (voir annexe II). Cette approximation est considérée acceptable sauf si $k = 3$ et que $m_i \leq 5$, $i = 1,2,...,k$. Suite à

l'utilisation de l'approximation par la distribution du χ^2, si la probabilité est inférieure au seuil critique α, l'hypothèse H_0 est rejetée sinon elle est acceptée.

Toutefois, ce test ne permet pas d'évaluer plusieurs hypothèses simultanément, mais bien de comparer plusieurs échantillons simultanément sous une seule hypothèse. Ainsi, si l'hypothèse nulle est rejetée, cela signifie que nous avons suffisamment d'évidences pour croire qu'au moins un échantillon est significativement différent d'un autre. La portée de cette conclusion est bien limitée et ne nous permet pas de savoir de quel(s) échantillon(s) provient la différence. Pour connaître cette information, nous devons utiliser une procédure de comparaison multiple.

2.3.3 Procédure de comparaison multiple

Lorsque nous voulons évaluer plusieurs hypothèses simultanément, nous faisons face à un problème de multiplicité des tests. Il n'est pas approprié d'évaluer plusieurs hypothèses en effectuant plusieurs tests d'hypothèse à un taux α. Une telle pratique résulterait en un accroissement important du risque d'erreur α proportionnel au nombre de tests. Lorsque plusieurs hypothèses doivent être évaluées en même temps, le contrôle du risque d'erreur α est transposé à un contrôle du risque d'erreur global (FWER *Family Wise Error Rate*) qui est le risque de faire une erreur de type I parmi tous les tests d'hypothèses. Cela résulte nécessairement en une perte de puissance statistique mais protège la rigueur des résultats.

Les méthodes de comparaisons multiples (MCP), qui contrôlent le FWER nous permettront de comparer simultanément les performances de plusieurs processus stochastiques différents par paire, en s'assurant d'obtenir un niveau de confiance statistique contrôlé. Par exemple, pour k échantillons, nous pourrons effectuer les $\left(k\left(k-1\right)\right)/2$ comparaisons possibles par paire d'échantillons.

Un test de comparaison multiple doit être fait uniquement lorsque l'hypothèse nulle a été rejetée, permettant ainsi de découvrir la source du rejet de H_0. Suite à un test de Kruskal-Wallis, Hochberg & Tamhane (1987) recommande d'utiliser le *Tukey's honestly significant difference criterion* (HSD). Le HSD représente la deuxième étape de notre procédure de comparaison des performances, lorsqu'une différence est détectée.

2.3.3.1 Test HSD

Ici aussi, considérons que nous voulons comparer k échantillons distincts, possédant chacun m_i observations où $i = 1,2,...,k$. Les échantillons i et j sont différents si l'équation (2.8) est vraie, sinon, nous pouvons conclure que les deux échantillons proviennent de la même distribution.

$$\left|\bar{R}_i - \bar{R}_j\right| \leq \frac{Q_{k,\infty}^{(\alpha)}}{\sqrt{2}}\sqrt{d_{ij}} \quad i,j<k \tag{2.8}$$

Sachant que

$$d_{ij} = \left(\frac{M(M+1)}{12} - \frac{\sum t^3 - t}{12(M-1)}\right)\left(\frac{1}{m_i} + \frac{1}{m_j}\right) \tag{2.9}$$

Aussi, $Q_{k,\nu}^{(\alpha)}$ est le quantile de la loi de l'étendue Studentisée avec k échantillons et ν degrés de liberté (voir Annexe I).

2.3.3.2 Représentation graphique du test HSD

Le test HSD permet de déterminer les liens entre les groupes deux à la fois. Toutefois, la simple lecture des résultats numériques de ce test peut-être ardue. C'est pourquoi, ce test est généralement appuyé par une représentation graphique permettant d'évaluer simultanément

tous les résultats rapidement. Les intervalles de chaque échantillon peuvent être obtenus à l'aide des équations (2.9), (2.10) et (2.11).

$$\Delta = \frac{Q_{k,\infty}^{(\alpha)}}{\sqrt{2}} w_i \tag{2.10}$$

$$w_i = \frac{(k-1)\sum_{j=1}^{k}\sqrt{d_{ij}} - \sum_{j=1}^{k-1}\sum_{l>j}^{k}\sqrt{d_{jl}}}{(k-1)(k-2)} \tag{2.11}$$

De telle sorte que l'intervalle de l'échantillon i correspond à l'équation (2.12) suivante :

$$\bar{R}_i \pm \Delta \tag{2.12}$$

La figure (2.1) est un exemple typique d'une représentation graphique d'un test de comparaison multiple. Chaque échantillon est représenté par un trait. La longueur du trait est déterminée par le nombre d'observations provenant de cet échantillon. Plus le nombre d'observations est élevé, plus l'incertitude est réduite et donc plus la longueur du trait sera réduite aussi.

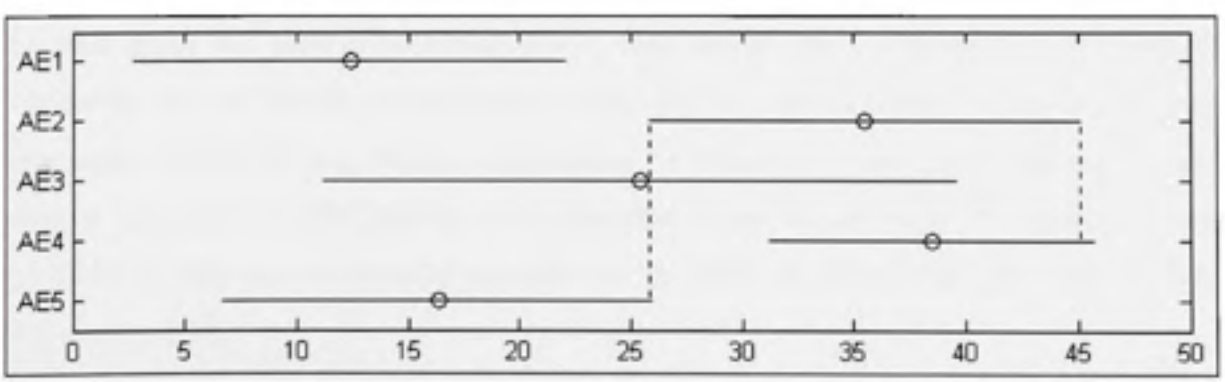

Figure 2.1 ***Exemple d'une comparaison multiple entre cinq échantillons.***

Sur la figure (2.1), les traits des échantillons AE1, AE2 et AE5 ont tous la même longueur, donc le même nombre d'observations, tandis que l'échantillon AE3 possède moins d'observations et que l'échantillon AE4 en possède plus. L'axe des abscisses représente les rangs (dans le cas de Kruskal-Wallis). Le centre de chaque trait, identifié d'un cercle représente le rang moyen de chaque échantillon. Toutefois, dans notre cas, comme nous avons utilisé une statistique de rang, l'intervalle représenté par les traits ne peut être utilisé que pour établir la relation entre les échantillons, mais ne représente pas des intervalles de confiance. Deux échantillons peuvent être considérés statistiquement différents sous un risque contrôlé α s'ils ne se superposent pas. Si les deux traits possèdent une valeur commune, nous ne possédons pas suffisamment d'évidence (par rapport à α) permettant d'affirmer qu'ils proviennent de deux distributions différentes. Ainsi, dans l'exemple présent, A1 est différent de A2 et A4, A2 est uniquement différent de A1, A3 n'est pas différentiable des autres échantillons, A4 se différentie de A1 et A5 et finalement, A5 ne se distingue que de A4. De cette observation, nous pourrions tirer comme conclusion que la distribution la plus performante est soit AE1, AE3 ou AE5. Nous pourrions être tentés de dire que AE1 est la meilleure solution mais cette hypothèse, en considérant le graphique précédent compromettrait le risque d'erreur du test.

2.3.4 Qualité d'une instance

Le plus grand défi dans l'élaboration d'une méthodologie de comparaison statistiquement rigoureuse et non biaisée, réside dans la création d'un unique critère de qualité Q, sans introduire de biais ni trop limiter la puissance de l'analyse. L'objectif est de trouver une mesure qui permet de différencier et de quantifier toutes les instances. Par exemple, il est possible de faire une sommation pondérée de la valeur d'optimisation ϕ et de η selon l'équation (2.13).

$$Q(N_{\max}) = a \cdot \phi\left(\eta(N_{\max})\right) + b \cdot \eta(N_{\max}) \quad a,b > 0 \tag{2.13}$$

La partie $a \cdot \phi(\eta(N_{max}))$ de l'équation(2.13) permet de différentier les instances n'ayant pas atteint l'optimum tandis que la partie $b \cdot \eta(N_{max})$ permet de différentier les instances ayant convergé sur l'optimum. Toutefois, en procédant de cette manière, le choix arbitraire des pondérations a et b aura des répercussions directes sur les distributions de Q des échantillons possédant à la fois des observations ayant atteint l'optimum et d'autres ne l'ayant pas atteint. À quel point l'atteinte de l'optimum doit-elle bonifier la qualité d'une solution? La réponse à cette question est très subjective mais le rang relatif des instances ne l'est pas, et il est tout à fait indépendant de a et b tel que $a,b>0$. Au delà des réelles valeurs de Q, l'équation (2.13) prend tout son sens lorsque l'on considère le rang des individus. Cette équation permet d'attribuer des valeurs de qualités ordonnées qui pourront ensuite recevoir le rang qui leur revient et ce, sans égard aux pondérations.

Considérons M individus devant être classés en rang où après N_{max} évaluations, I instances $(0 \le I \le M)$ auraient convergé et $M-I$ instances n'auraient pas atteint l'optimum. Il est évident que les I instances ayant convergé possèderont les I premiers rangs et que les $M-I$ autres obtiendront les $M-I$ derniers rangs. Il ne reste donc plus qu'à faire un classement intergroupe. Parmi les I instances ayant convergé, la vitesse de convergence permet de classer les individus. Celui ayant convergé le plus rapidement reçoit le premier rang tandis que celui ayant convergé le plus lentement, obtient le dernier rang, soit le rang I. Ensuite, des instances n'ayant pas convergé, celle possédant la meilleure valeur de coût du groupe reçoit le rang $I+1$ et ainsi de suite jusqu'à l'individu ayant la pire valeur de coût recevant le rang M.

L'équation (2.13) permet d'attribuer un tel ordonnancement. Les I instances ayant convergé auront toutes la même valeur de ϕ, plus petite que les $M-I$ instances n'ayant pas convergé et leur valeur de η sera aussi plus petite ou égale à celle des $M-I$ instances inférieures. La répartition des rangs parmi les I instances supérieures est basée uniquement sur les valeurs

η de ceux-ci tandis que le classement des $M-I$ instances d'adaptation moindre sont classées selon leur niveau d'adaptation $\phi(N_{max})$.

2.3.5 Analyse globale

La méthode proposée jusqu'ici permet de comparer des processus d'optimisation stochastiques pour une seule valeur de N_{max}. Pour comparer plusieurs valeurs de N_{max}, il suffit de reprendre le processus de comparaison pour différentes valeurs temporelles. Idéalement, le but est de connaître les résultats des comparaisons pour toutes les valeurs de N_{max}. Par contre, comme énoncé aux sections 2.1.1.1 et 2.1.1.3, nous sommes dans l'obligation de limiter l'étude à une fenêtre d'analyse finie et échantillonnée.

2.3.5.1 Réduction du nombre de comparaisons

L'utilisation du rang des observations plutôt que leur valeur réelle nous permet de limiter considérablement le nombre de comparaisons à faire. Effectivement, nous pouvons faire des comparaisons uniquement pour les valeurs de N_{max} pour lesquelles le rang moyen d'au moins deux échantillons est différent de celle obtenues pour le paramétrage temporel précédemment échantillonné. Cela simplifie grandement l'analyse globale en limitant sensiblement le nombre de comparaisons à faire.

2.3.5.2 Affichage graphique

La figure 2.1 est une représentation graphique d'une comparaison multiple permettant de simplifier l'analyse. Pour représenter graphiquement les résultats de comparaisons répétées sur une fenêtre temporelle, une nouvelle méthode d'affichage est proposée. Il s'agit simplement de superposer tous les traits des intervalles de confiance et d'attribuer une couleur distincte pour chaque échantillon, puis de refaire la procédure pour chaque paramétrage temporel comme démontré sur la figure 2.2.

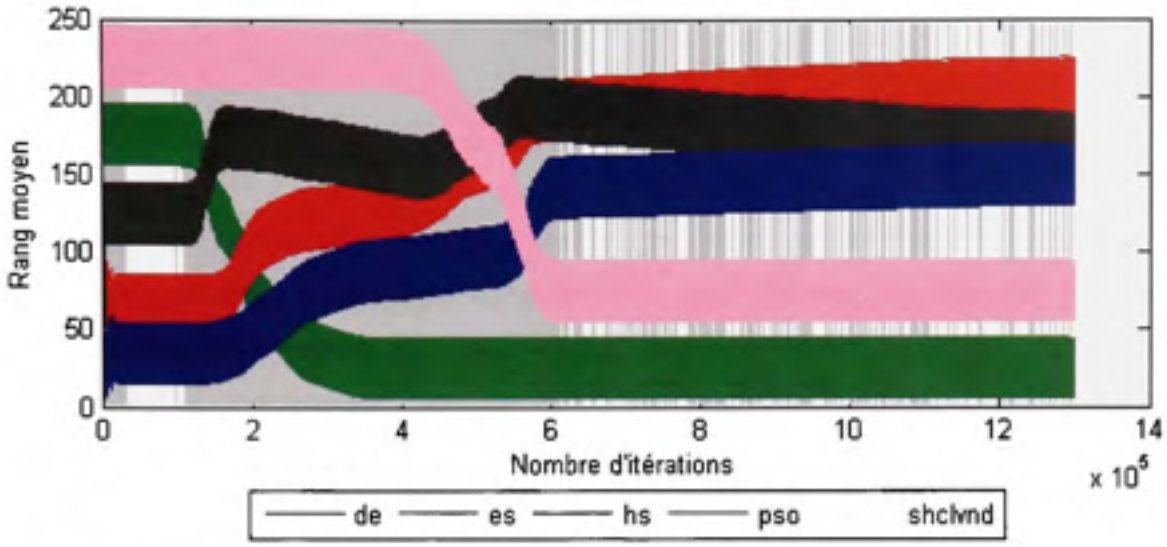

Figure 2.2 ***Comparaison multiple de cinq algorithmes étendue sur un domaine temporel.***

L'analyse de la figure fonctionne de la manière suivante : l'axe des x représente le paramétrage temporel N_{max} et l'axe y est le rang moyen des échantillons. Chaque échantillon est représenté par une bande de couleur couvrant une plage complète de paramétrage temporel. L'épaisseur des traits constitue l'intervalle de confiance des échantillons au même titre que la largeur des traits de la figure2.1. On remarque également en arrière plan des zones grisées. Ces zones représentent les itérations pour lesquelles il y a eu un changement de rang et donc une comparaison statistique d'effectuée.

CHAPITRE 3

DÉFINITION DU CADRE EXPÉRIMENTAL – LES ALGORITHMES

Le présent chapitre permettra de définir un cadre expérimental complet pour l'évaluation des performances de cinq algorithmes d'optimisation bien connus. En premier lieu, une description exhaustive de chacun des algorithmes sera faite. Ensuite, la définition d'un banc de test et de ses caractéristiques sera effectuée. Finalement, le protocole expérimental sera développé. Avant tout, comme les cinq algorithmes étudiés sont des algorithmes évolutionnaires, voyons ce qui caractérise ce type d'algorithme.

3.1 Les algorithmes

3.1.1 Les algorithmes évolutionnaires

Les algorithmes évolutionnaires (AEs) représentent une sous-classe des métaheuristiques possédant des caractéristiques particulières rappelant la théorie darwinienne de l'évolution des espèces. Plus précisément, la structure des AEs prend la forme de générations de population dont l'évolution se caractérise par des opérateurs de sélection et de reproduction des individus des populations, auxquels s'ajoute un opérateur de mutation. Nous noterons $\vec{X} = \{\vec{x}_1, \vec{x}_2, ..., \vec{x}_p\}$ une population contenant p solutions.

L'utilisation d'une population de solution, plutôt que d'une seule solution est un mécanisme qui permet d'éviter, dans une certaine mesure, de rester prisonnier d'un minimum local puisque le bassin de solutions permet de fouiller simultanément plusieurs régions. Par contre, ce principe possède certaines limites de sorte qu'il est difficile de créer des algorithmes qui évoluent rapidement tout en évitant l'effondrement des individus d'une population vers une solution unique. La figure 3.1 représente le cycle d'évolution d'un algorithme évolutionnaire typique.

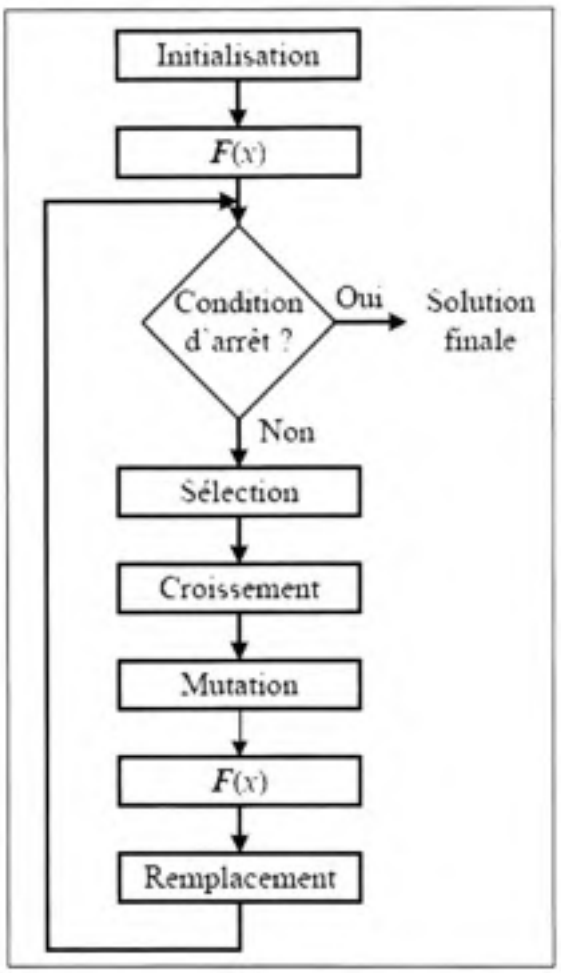

Figure 3.1 ***Cycle d'évolution d'un AE***

Ce cycle est une émulation très simplifiée du mécanisme d'évolution des espèces et représente le cycle de création et d'évolution d'une population complète. L'initialisation consiste à générer une toute première population de solutions. Généralement, la population initiale est créée de manière purement aléatoire. Il est aussi possible que des informations *a priori* nous permettent de croire que certaines régions du domaine Ω soient plus prometteuses que d'autres, il est possible d'en prendre avantage lors du processus de création de la population initiale. Par contre, il n'est pas utile de développer des algorithmes d'initialisation trop complexes qui pourraient limiter les mécanismes d'évolution de l'algorithme. La valeur de coût représente le niveau d'adaptation par rapport à l'environnement(f), dans lequel se situe chaque individu de la population. Les techniques

de sélection imitent la pression de l'environnement en favorisant la reproduction des solutions les plus prometteuses, c'est-à-dire celles dont l'adaptation est la meilleure. Les techniques de reproduction permettent quant à elles, de générer des nouvelles solutions possédant des caractéristiques performantes des solutions sélectionnées. La mutation permet d'introduire une part de hasard permettant d'ajouter de nouvelles caractéristiques susceptibles d'améliorer l'individu et ultimement les prochaines générations.

Chaque algorithme évolutionnaire implémente d'une manière plus ou moins rigoureuse ce cycle auquel peuvent se greffer d'autres mécanismes. Chaque algorithme possède son propre système de mutation et de croisement. Dans certains cas, ces deux opérations sont scindées en une seule et de ce cycle de base découle une multitude de variantes.

Il est important de noter ici qu'une itération correspond bien à l'échantillonnage d'une seule solution et non à l'échantillonnage d'une population. Par exemple, pour un AE ayant une population de 20 individus, chaque génération serait constituée de 20 itérations. Cette unité de mesure est beaucoup plus juste pour établir un rapport impartial entre les temps de calcul de plusieurs AEs ayant des populations différentes.

3.1.2 Paramétrage des algorithmes

Le paramétrage d'un algorithme évolutionnaire, ou plus globalement d'une métaheuristique, est un facteur important de sa performance pour un problème donné et peut parfois être aussi crucial et problématique que le choix d'un algorithme. En réalité, les différents paramétrages possibles d'un algorithme sont autant d'algorithmes différents ayant certes des similitudes importantes mais possédant également des propriétés propres qui influencent leur comportement. À ce titre, il est aussi justifié de comparer les performances d'algorithmes différents que de comparer différents paramétrages du même algorithme.

Dans le cadre de ce travail, le choix du paramétrage des algorithmes étudiés est principalement basé sur leur forme canonique. Par contre, suite à un certain nombre d'essais,

il est devenu évident que certaines formes canoniques étaient très mal adaptées, donnaient de très mauvais résultats et étaient un piètre reflet des réelles capacités de l'algorithme. L'ajout de certains mécanismes issus de travaux plus récents a permis de rendre les performances des algorithmes plus homogènes et plus représentatives de leurs capacités actuelles. Notons qu'aucune modification qui aurait volontairement abaissé la performance d'un algorithme n'a été faite. Aussi, comme l'objectif de ce travail ne portait pas directement sur les algorithmes d'optimisation, il a été établi dès le départ qu'une seule version et un seul paramétrage de chaque algorithme seraient utilisés pour tous les problèmes du banc de test. Il est évident que pour un banc de test donné constitué de plusieurs problèmes ayant des propriétés très différentes, un unique calibrage ne permet fort probablement pas d'obtenir les meilleurs résultats possibles de l'algorithme sur tous les problèmes. Toutefois, l'objectif de ce travail est de proposer une méthodologie de comparaison générique des performances d'algorithmes non déterministes itératifs. Dans cette optique, le choix d'algorithmes à comparer fait office de prétexte permettant d'évaluer la méthodologie plutôt que de comparer les algorithmes sous leurs formes optimales. Il est donc évident que d'autres calibrations ou encore d'autres versions des algorithmes implémentés pourraient donner de meilleurs résultats. Cela n'enlève toutefois rien aux résultats obtenus dans la mesure où ils dressent un portrait fidèle des performances d'une implémentation particulière des algorithmes pour une fenêtre temporelle élargie.

Il est aussi intéressant de noter que les processus d'évolution des cinq algorithmes implémentés ici ne sont pas affectés par les bornes $\vec{L}$ et $\vec{U}$ (définies au chapitre 1). Les bornes ne sont effectives que lors de l'initialisation. Lors de l'évolution, rien n'empêche les algorithmes d'explorer à l'extérieur des bornes.

3.1.3 Notations utilisées

Les métaheuristiques font appel à différentes distributions de probabilités. Dans notre cas, trois distributions seront utilisées, soit la distribution uniforme, la distribution uniforme

discrète et la distribution normale, ou gaussienne. Les notations de chacune de ces distributions sont les suivantes :

- Distribution uniforme continue sur l'intervalle [a,b] : $U(a,b)$

- Distribution uniforme discrète sur l'intervalle [a,b], dans notre cas, cette loi permet de tirer au hasard parmi une liste d'index : $D(a,b)$ où $D(a,b) = a + \lfloor U(0,1) \cdot (b - a + 1) \rfloor$

- Distribution normale (gaussienne) centrée sur μ et d'écart-type σ : $N(\mu,\sigma)$

Aussi, pour certains algorithmes, la moins bonne solution d'un ensemble doit être connue. Le symbole « # » en exposant permet d'identifier cette pire solution au même titre que « * » permet d'identifier la meilleure solution.

3.1.4 Stratégie d'évolution

La stratégie d'évolution (*Evolution Strategy* - ES) Rechenberg (1965) constitue la toute première métaheuristique et fut développée en 1965 par trois étudiants de l'institut Hermann Föttinger de l'université technique de Berlin. À l'époque, les travaux de ces trois étudiants portaient sur l'aérodynamisme de différents objets dans des milieux turbulents. En 1964, ils créèrent un système de test permettant d'étudier l'aérodynamisme de différentes configurations d'une plaque flexible bidimensionnelle. La plaque était positionnée à la sortie d'un tunnel de vent et était itérativement remodelée et évaluée, tout ça dans le but d'obtenir une forme optimale. Le remodelage de la pièce était obtenu à l'aide d'un robot suivant un ensemble de règles. Dans ce contexte, la forme la plus optimale est celle où la pièce est complètement plate et parallèle au jet d'air et pourtant, les méthodes basées sur le gradient donnaient toujours une forme en "S", un minimum local. Ils démontrèrent alors qu'un

ensemble de règles issues d'une simple heuristique donnait de meilleurs résultats qu'un algorithme basé sur le gradient.

Klockgether & Schwefel (1970) obtinrent des résultats encore plus intéressants en appliquant l'algorithme au problème de la forme de la sortie du système de propulsion d'un jet. Ils arrivèrent à une forme très contre intuitive qui était pourtant beaucoup plus efficace que la forme connue donnant les meilleures performances. Une meilleure compréhension des phénomènes impliqués permit un peu plus tard de comprendre l'optimalité de cette forme. Le jeu de règles était très simple et suivait seulement deux règles. À chaque itération :

- Toutes les variables doivent changer légèrement de valeur, de façon aléatoire.
- Si la nouvelle solution n'est pas moins bonne que l'ancienne, on conserve cette solution sinon, on revient à l'ancienne solution.

En parallèle à la théorie de l'évolution, la première règle est une mutation tandis que la seconde est la sélection des meilleurs individus, où seuls les meilleurs spécimens survivent. Il est intéressant de noter que cette forme de l'algorithme n'utilise qu'un seul individu à la fois plutôt qu'une population de plusieurs individus. Rechenberg (1971) utilisa ensuite des distributions gaussiennes comme outils de mutation. Ainsi, en supposant une solution quelconque $\vec{x}$, la création d'une nouvelle solution $\vec{x}'$ suit l'équation (2.14) suivante :

$$\vec{x}' = \vec{x} + N(0, \vec{\sigma}) \tag{2.14}$$

Ici toutefois, le vecteur écart-type $\vec{\sigma}$ est le seul paramètre permettant d'ajuster l'étendue de l'exploration, l'amplitude de la mutation. Une petite valeur de $\vec{\sigma}$ favorise une exploitation locale tandis qu'une grande valeur, permet plutôt d'explorer une plus grande surface. L'adaptation de ce paramètre au cours de l'évolution est primordial pour cette forme de ES. Une valeur fixe de $\vec{\sigma}$ permet une évolution efficace des premières générations mais limite rapidement l'évolution de l'algorithme. Rechenberg (1973) proposa alors la fameuse règle du 1/5. Cette règle permet d'adapter dynamiquement $\vec{\sigma}$ selon le taux de succès des mutations.

Rechenberg (1973) démontre que pour obtenir une évolution maximale de la forme canonique de l'algorithme, $\vec{x}'$ doit être en moyenne un fois sur cinq meilleure que $\vec{x}$. Cela signifie qu'au cours du processus d'optimisation, $\vec{\sigma}$ doit être réajusté de manière à contrôler le taux de succès P_s pour qu'il soit maintenu autour de 1/5. La procédure est simple et consiste à prédéterminer un pas d'adaptation G de $\vec{\sigma}$. Considérons G_s comme étant le nombre de mutants ayant été supérieur à son ancêtre. Le taux de succès P_s est défini à l'aide de l'équation (2.15).

$$P_s = \frac{G_s}{G} \tag{2.15}$$

Puis l'ajustement de $\vec{\sigma}$ se fait à l'aide de l'équation (2.16) suivante :

$$\vec{\sigma} = \begin{cases} \vec{\sigma}/\alpha & \text{si } P_s > 1/5 \\ \vec{\sigma} \cdot \alpha & \text{si } P_s < 1/5 \\ \vec{\sigma} & \text{si } P_s = 1/5 \end{cases} \tag{2.16}$$

Plus de quarante ans après l'avènement de cette nouvelle branche d'optimisation, de nombreuses variantes de cet algorithme ont vu le jour. Il existe une nomenclature simple permettant de définir les principales caractéristiques de la version utilisée. La version canonique, utilisée dans le cadre de ce travail est la forme $(1+1)-\mathrm{ES}$. D'une façon générique, les différentes versions de ES s'écrivent sous la forme $(\mu/\rho\{+,\}\lambda)-\mathrm{ES}$. Où μ correspond au nombre de parents permettant de recréer λ enfants. La génération d'un seul enfant est faites depuis la recombinaison de ρ parents $(\rho \leq \mu)$. Les symboles "+" et "," permettent de connaître le mode de sélection. Le "+" signifie que les enfants et les parents peuvent être sélectionnés pour populer la prochaine génération tandis que la version "," ne permet pas aux parents de survivre même s'ils sont de meilleure qualité que leurs enfants. Lorsque ρ est supérieur à 1, l'algorithme utilise des méthodes de recombinaisons d'individus. La forme canonique utilise une distribution isotropique. Cela signifie que

chacune des composantes de $\vec{\sigma}$ est adaptée à l'aide d'un coefficient unique et que toutes les composantes sont donc intimement liées. Il existe des versions non-isotropiques qui permettent de dissocier les dimensions de $\vec{\sigma}$ selon l'évolution des solutions. Cela permet de mieux suivre la structure du problème à optimiser.

3.1.4.1 Notre implémentation

Dans le présent document, la forme canonique $(1+1)-\text{ES}$ à distribution isotropique avec adaptation selon la règle du 1/5 été implémentée.

Algorithme 1 ES

Arguments :
Ψ : Constante d'étalement initial
G : Pas d'adaptation
α : Amplitude d'adaptation

1: $\vec{\sigma} = \left(\vec{H} - \vec{L}\right) \cdot \Psi$
2: **tant que** *critère d'arrêt pas atteint* **faire**
3: $G_s = 0$
4: **pour** $i := 1$ à G **faire**
5: $\vec{x}' = \vec{x} + \vec{N}(0, \vec{\sigma})$
6: **si** $f(\vec{x}') < f(\vec{x})$ **alors**
7: $G_s = G_s + 1$
8: $\vec{x} = \vec{x}'$
9: **fin si**
10: **fin pour**
11: **si** $G_s/G < 1/5$ **alors**
12: $\vec{\sigma} = \vec{\sigma}/\alpha$
13: **sinon si** $G_s/G > 1/5$ **alors**
14: $\vec{\sigma} = \vec{\sigma} \cdot \alpha$
15: **fin si**
16: **fin tant que**

Pour $d \geq 30$, Beyer & Schwefel (2002) conseillent d'utiliser $G = d$ et sous cette condition, $0.85 \leq \alpha < 1$. Des essais préliminaires permirent de constater que cette règle était efficace même pour $d < 30$. Ainsi, dans le cadre de ce travail, il a été défini que $G = d$ et $\alpha = 0.98$.

Aussi, l'initialisation de l'étalement est dépendante du domaine de fouille. L'utilisation d'une règle simple (ligne 1 du pseudo-code ES) permet de s'assurer qu'à l'initialisation, les premières générations puissent couvrir l'ensemble du domaine de fouille borné (Rudolf & Köppen, 1996) où $\Psi = 0.5$.

3.1.5 Évolution différentielle (DE)

L'évolution différentielle de Storn & Price (1995) est un algorithme d'optimisation qui a connu énormément de succès depuis son apparition et qui fut initialement créé pour résoudre des problèmes continus. Soit une population de solutions, aléatoirement initialisées constituée de p individus.

Le processus d'évolution d'une génération suit un cycle simple permettant d'améliorer séquentiellement chacun des p individus. Ainsi, à tour de rôle chacun des p individus est appelé à être le vecteur cible. Un mutant est créé à l'aide d'un processus de mutation qui consiste simplement à ajouter la différence pondérée de deux autres individus de la population à un tiers vecteur comme démontré par l'équation (2.17).

$$v = x_a + c_1 \left(x_b - x_c \right) \quad a \neq b \neq c \tag{2.17}$$

Dans l'équation précédente, le coefficient c_1 est le coefficient de mutation qui permet de contrôler l'amplitude des mutations. Un processus de croisement discret permet ensuite de créer une nouvelle solution en croisant le vecteur mutant v nouvellement créé au vecteur cible. Un taux de croisement cr permet également de gérer le niveau d'implication du vecteur cible et du vecteur mutant dans la création de la nouvelle solution. Le croisement permet tout simplement d'assigner à chacun des d paramètres de la nouvelle solution la valeur du vecteur mutant ou du vecteur cible x_i (équation (2.18)).

$$x_i' = \begin{cases} x_i & \text{si } U_i(0,1) > cr \ \wedge \ D(0,p) \neq \text{i} \\ v_i & \text{sinon} \end{cases} \tag{2.18}$$

Dans l'équation (2.18), l'ajout de la condition $D(0,p) \neq i$ permet d'éviter de créer des clones en s'assurant que les nouvelles solutions aient au moins une dimension issue du vecteur mutant. Ensuite, une sélection permettra de choisir la meilleure des deux solutions entre x_i et x_i'. En observant bien les étapes de mutation et de croisement il est évident que pour être fonctionnel, le nombre d'individus de la population doit être au minimum 4.

Il existe une notation particulière de l'algorithme, comme pour la stratégie d'évolution, qui permet de noter la forme de l'algorithme utilisé. La notation est la suivante : DE/x/y/z. La variable x fait référence au mode de sélection du vecteur de base x_a pour la mutation. Dans notre cas, la sélection est purement aléatoire (rand) mais il existe également deux autres versions permettant de favoriser le meilleur vecteur (best), (rand-to-best). La variable y détermine le nombre de différentiations utilisées lors de la mutation. Dans notre cas, nous en avons utilisé une seule alors qu'il est possible d'en utiliser plus. Finalement, la variable z représente le mode de croisement. Dans le cas étudié, le mode de croisement est discret (bin), mais il existe aussi un autre mode (exp).

3.1.5.1 Notre implémentation

Nous avons implémenté la forme DE/rand/1/bin de l'algorithme. Le paramétrage de cet algorithme n'est pas clairement défini. Toutefois, Storn suggère d'utiliser $p = 10d$, qui s'est avéré très performant. En utilisant une simple règle plutôt qu'une valeur fixe, peu importe le nombre de dimensions, l'algorithme est dès le départ un peu mieux adapté aux problèmes. Aussi, des essais ont permis de constater que l'utilisation d'une très petite valeur de cr permettait de limiter que l'algorithme soit piégé dans un optimum local sans trop ralentir l'évolution. Une valeur de $cr = 0.1$ a ainsi permis d'obtenir des résultats très intéressants.

Finalement, le coefficient d'amplitude des mutations c_1 a été ajusté à 0.8, comme suggéré par Storn.

Algorithme 2 DE

Arguments :
cr : Rapport d'implication
c_1 : Amplitude de mutation
p : Nombre d'individus

tant que *critère d'arrêt pas atteint* **faire**
  pour $i = 1$ à p **faire**
    $a = D_1(0, p) \wedge \neq i$
    $b = D_2(0, p) \wedge \neq i \wedge \neq a$
    $c = D_3(0, p) \wedge \neq i \wedge \neq a \wedge \neq b$
    $\vec{v} = \vec{x}_a + c_1 \cdot (\vec{x}_b - \vec{x}_c)$
    $\Theta = D_4(0, d)$
    pour $j = 1$ à d **faire**
      si $U_1(0,1) > cr \wedge \Theta \neq j$ **alors**
        $x'_j = x_{i,j}$
      sinon
        $x'_j = v_j$
      fin si
    fin pour
    si $f(\vec{x}'_i) < f(\vec{x}_i)$ **alors**
      $\vec{X} = (\vec{X} \setminus \{\vec{x}_i\}) \cup \{\vec{x}'\}$
    fin si
  fin pour
fin tant que

3.1.6 Fouille par harmonie (HS)

La fouille par harmonie (*Harmony Search*, HS) Geem & al. (2001) est une métaheuristique inspirée par la recherche d'une meilleure harmonie musicale lors d'une session de musique improvisée. Une harmonie musicale est une notion purement esthétique lors d'une combinaison de plusieurs sons simultanés, et au même titre qu'un algorithme d'optimisation tend à obtenir une configuration idéale parmi les états de plusieurs variables selon une

fonction d'optimisation donnée, un groupe de musiciens recherche une harmonie parfaite en modulant l'état de chacun des instruments impliqués.

La notion d'exploitation de cet algorithme se fait à l'aide d'une base de données, *HM* (*Harmony Memory*) constituée des *HMS* (*Harmony Memory Size*) solutions les plus performantes échantillonnées.

$$HM = \begin{bmatrix} HM_{1,1} & HM_{1,2} & \cdots & HM_{1,d} \\ HM_{2,1} & HM_{2,2} & \cdots & HM_{2,d} \\ \vdots & \vdots & \cdots & \vdots \\ HM_{HMS,1} & HM_{HMS,2} & \cdots & HM_{HMS,d} \end{bmatrix}$$

Toutefois, il est important de mentionner ici que la mémoire *HM* ne possède aucun mécanisme permettant d'éviter les clones. Il est donc possible (voir très probable) que plusieurs copies d'une même solution se retrouvent simultanément dans la mémoire *HM* du processus.

Une des particularités de cet algorithme est que le processus générateur construit les nouvelles solutions, dimension par dimension. La création de chaque dimension d'une nouvelle solution est indépendante des autres dimensions. De plus, le processus générateur est constitué de trois mécanismes différents dont la probabilité de réalisation de chacun d'eux est définie par l'utilisateur à l'aide de deux paramètres. Le *HMCR* (*Harmony Memory Considering Rate*), comme son nom l'indique, permet de déterminer si la mémoire *HM* sera utilisée. Lorsque la mémoire *HM* est utilisée, un second paramètre, le *PAR* (*Pitch Adjusting Rate*) permet de générer, ou non, une légère variation additive autour d'un élément de *HM* . Lorsque la mémoire n'est pas utilisée, le mécanisme de création est purement aléatoire sur la totalité du domaine de fouille de la dimension en question. Ces trois processus permettent ainsi de coupler l'exploitation des solutions antérieures à l'exploitation des nouvelles composantes selon une probabilité prédéfinie.

L'initialisation de cet algorithme consiste simplement à remplir aléatoirement la mémoire *HM* de *HMS* solutions distinctes. Considérons maintenant la création d'une nouvelle solution $\vec{x}'$. Soit les variables aléatoires $U_1(0,1)$, $U_2(0,1)$, $U_3(-1,1)$ et $D(1,HMS)$ et la dimension j de $\vec{x}'$ à générer. La première variable aléatoire permet de déterminer si la dimension j de la solution sera produite à partir de la mémoire ou si elle sera créée de manière purement aléatoire. La probabilité de réalisation de ces deux possibilités est déterminée par le taux *HMCR*. Si la mémoire est choisie, alors la dimension prendra la valeur de celle d'une des *HMS* solutions mémorisées. Finalement, si $U_2(0,1) \leq PAR$, une valeur aléatoire équivalente à $\alpha \cdot U_3(-1,1)$ sera ajouté à la valeur déjà donnée.

3.1.6.1 Notre implémentation

Tel qu'utilisé par Lee & Geem (2004), le nombre de solutions mémorisées (*HMS*) a été fixé à 10, le taux de considération de la mémoire *HMCR* a été fixé à 0.85, le taux d'ajustement local a été fixé à 0.45 et finalement, comme aucune indication particulière n'est faite à propos du coefficient d'impact α, celui-ci a été fixé à 1.

Algorithme 3 HS

Arguments :

$HMCR$: Taux de considération de la mémoire
PAR : Taux d'ajustement local
HMS : Dimension de HM
HM : Mémoire d'harmonie
α : coefficient d'impact de PAR

tant que *critère d'arrêt pas atteint* **faire**
  pour $j = 1$ à d **faire**
    si $U_1(0,1) \leq HMCR$ **alors**
      $\delta = D_1(0, HMS)$
      $x_j^{'} = HM_{\delta,j}$
      si $U_2(0,1) \leq PAR$ **alors**
        $x_j^{'} = x_j^{'} + \alpha \cdot U_3(-1,1)$
      fin si
    sinon
      $x_j^{'} = U_4(L_j, H_j)$
    fin si
  fin pour
  si $f(\vec{x}^{'}) < f(\vec{x}^{a})$ **alors**
    $HM = (HM \setminus \{\vec{x}^{a}\}) \cup \{\vec{x}^{'}\}$
    Réajuster $\vec{x}^{a}$
  fin si
fin tant que

3.1.7 Optimisation par essaim de particules (PSO)

L'optimisation par essaim de particules (*Particles Swarm Optimization*, PSO) Kennedy & Eberhart (1995) est une méthode d'optimisation qui diffère sensiblement des quatre autres algorithmes d'optimisation comparés dans le cadre de ce travail. Contrairement aux autres méthodes citées qui évoluent par compétition entre les individus en utilisant des opérateurs de croisements et de mutations, cette méthode utilise plutôt la coopération entre les individus. Cette méthode, initialement conçue pour l'optimisation de problèmes continus, est une métaphore simplifiée des comportements sociaux que l'on retrouve au sein de groupes d'animaux tels les bancs de poissons, les volées d'oiseaux ou les essaims d'insectes. Ce qui caractérise ces groupes est le concept d'intelligence commune où chaque individu ne possède qu'une information partielle de son environnement qui collectivement, permet de gérer

intelligemment le groupe. Chaque individu est influencé à la fois par ses propres expériences passées et par celles du groupe. Malgré les différences notables entre le principe directeur de cet algorithme et celui des algorithmes évolutionnaires en général, l'optimisation PSO fait tout de même partie de cette classe. Wilson (1975) démontra que le partage d'informations entre individus d'un même groupe était, dans certains cas beaucoup plus avantageux pour la survie de l'espèce que la compétition entre les individus. La coopération est ainsi un mécanisme évolutif au même titre que la sélection. De plus, les mécanismes de sélection, de croisement et de mutation sont bien présents mais sous une forme moins implicite.

Au départ, lors du développement de l'algorithme, Kennedy & Eberhart (1995) tentèrent de créer un synchronisme entre les déplacements des individus à l'aide d'une règle simple. Après une initialisation aléatoire de la position des individus et de leur vélocité (en vitesse et en direction), la règle était d'attribuer à chaque individu la vitesse et la direction de son plus proche voisin. Toutefois cette simple règle n'était pas suffisante et faisait en sorte que rapidement, toute la population convergeait sur une seule direction fixe. Cette faiblesse fut contournée par l'ajout d'un « facteur de folie », une composante stochastique ajoutée à chaque itération à la vélocité d'individus choisis aléatoirement. Jusque-là, les individus n'étaient pas guidés par un objectif commun réel mais se déplaçaient plutôt selon une règle artificielle (le facteur de folie) qui arrivait tout de même à reproduire l'esthétique des déplacements d'essaims. Heppner & Grenander (1990) simulaient aussi, mais dans un tout autre but, les déplacements d'une population d'oiseaux en utilisant un marqueur à atteindre par la volée d'oiseaux. Le résultat était très intéressant d'un point de vue esthétique mais considérait que les oiseaux connaissaient déjà la position de l'objectif, ce qui ne reflète pas l'objectif primaire de l'algorithme PSO. Par exemple lorsque l'on ajoute une mangeoire pour oiseau, un premier oiseau vient s'y nourrir et en peu de temps, plusieurs autres viennent. Un processus de partage d'informations parmi les individus de la bande se produit et c'est précisément ce processus que l'on souhaite reproduire.

Par la suite, Kennedy & Eberhart (1995) retirèrent le « facteur de folie » et la règle transmettant à un individu la vélocité de son plus proche voisin, par un système de partage

d'informations où la vélocité de chaque individu est guidée à la fois par la meilleure position visitée par lui-même et par la meilleure position visitée par l'ensemble du groupe. C'est la toute première version de l'algorithme PSO. Ainsi, soit la vélocité $\vec{v}_i$ d'un individu i actuellement à la position $\vec{x}_i$ pour lequel la meilleure position visitée par celui-ci est notée $\vec{x}_i^*$ et la meilleure position visitée par le groupe est notée $\vec{x}_g^*$. L'ajustement itératif de la vélocité suit l'équation (2.19) suivante :

$$\vec{v}_i = \vec{v}_i + c_1 \cdot \vec{U}_1(0,1)\left(\vec{x}_i^* - \vec{x}_i\right) + c_2 \cdot \vec{U}_2(0,1)\left(\vec{x}_g^* - \vec{x}_i\right) \tag{2.19}$$

La nouvelle position des particules est ensuite calculée à l'aide de l'équation(2.20) suivante :

$$\vec{x}_i = \vec{x}_i + \vec{v}_i \tag{2.20}$$

Évidemment, $\vec{x}_i^*$ et $\vec{x}_g^*$ sont réajustés à toutes les itérations et cela constitue un opérateur de sélection tout comme leur utilisation constitue un croisement. Finalement, les coefficients c_1, c_2 et les variables aléatoires U_1 et U_2 permettent les mutations. Une analyse détaillée de l'équation (2.19) permettra de mieux comprendre l'impact de chacune des composantes pour la gestion de la vélocité des particules.

$$\vec{v}_i = \underbrace{\vec{v}_i}_{A} + \underbrace{c_1 \cdot \vec{U}_1(0,1)\left(\vec{x}_i^* - \vec{x}_i\right)}_{B} + \underbrace{c_2 \cdot \vec{U}_2(0,1)\left(\vec{x}_g^* - \vec{x}_i\right)}_{C}$$

A. Cette composante permet d'ajouter de l'inertie aux particules de sorte que la vélocité précédente influence la suivante.

B. Cette composante permet d'ajouter l'historique de la particule en question dans la gestion de la vélocité. Le mouvement de la particule est « attiré » par la meilleure position visitée par la particule depuis l'initialisation de l'algorithme.

C. Cette composante permet d'ajouter l'historique de la population dans la gestion de la vélocité. Le mouvement de la particule est « attiré » par la meilleure position visitée par l'ensemble de la population depuis l'initialisation de l'algorithme.

L'impact de l'historique de la particule par rapport à l'impact de la population est géré à l'aide des coefficients c_1 et c_2. Ici, $c_1 = c_2 = 2$ de sorte que la moyenne soit de 1 pour chacun des facteurs.

3.1.7.1 Limite de la vélocité

Il arrive que la vélocité des particules s'emballe et devienne beaucoup trop grande pour la dimension du problème. Dans ce cas, les particules ne font plus une exploration efficace mais font plutôt des sauts d'un extrême à un autre sans s'améliorer. Dans ce cas, la vélocité des particules diverge et l'équilibre du système est perdu. Eberhart & Kennedy (1995) ajoutèrent alors une limite à la vélocité des particules. Puis, Clerc & Kennedy (2002) étudièrent en profondeur le comportement de l'algorithme et arrivèrent à mieux contrôler sa convergence à l'aide d'un facteur de constriction χ qui remplace plus efficacement et plus simplement la limite de vélocité. L'équation (2.19) devient alors l'équation (2.21) suivante :

$$\vec{v}_i = \chi\left[\vec{v}_i + \vec{U}_1\left(0,\frac{\varphi}{2}\right)\left(\vec{x}_i^{\,*} - \vec{x}_i\right) + \vec{U}_2\left(0,\frac{\varphi}{2}\right)\left(\vec{x}_g^{\,*} - \vec{x}_i\right)\right] \tag{2.21}$$

Pour préserver la convergence, Clerc & Kennedy (2002) et Kennedy (2003) suggèrent d'utiliser $\chi = 0.729$ et $\varphi = 4.1$.

3.1.7.2 Variantes importantes

L'optimisateur PSO est largement étudié et une multitude de variantes intéressantes ont vu le jour depuis sont apparition en 1995. Sans utiliser le facteur de constriction, Shi & Eberhart (1998) proposent d'utiliser un poids d'inertie w variable durant le processus. Initialement,

une grande valeur de w favorise une recherche globale en créant des plus grands déplacements, en limitant la précision de ces déplacements par l'amplification de l'inertie des particules tandis qu'une petite valeur, vers la fin, limite l'inertie et affine les déplacements de manière à favoriser une recherche locale. Une diminution linéaire de w durant la simulation pour améliorer l'algorithme est proposée. Commencer la recherche avec $w = 1$ pour terminer avec $w = 0.4$ donnerait de meilleurs résultats que la forme canonique de l'équation (2.19).

D'autres travaux étudièrent l'impact des topologies de voisinage entre les particules. Jusqu'ici, toutes les particules font partie d'un seul et unique groupe et sont toutes égales et communiquent toutes entre elles. Eberhart & Kennedy (1995) et Kennedy (1999) & Sugant han (1999) étudièrent l'effet de différentes topologies sociales.

3.1.7.3 Notre implémentation

Algorithme 4 PSO

Arguments :

χ : Facteur de convergence
φ : Facteur de convergence
p : Le nombre de particules

1. Initialiser la position $\vec{x}$ et la vélocité $\vec{v}$ des p particules
2. Pour chaque particule, $\vec{x}_i^* = \vec{x}_i$
3. $\vec{x}_g^*$ égale la meilleure position parmi toutes les particules
4. **tant que** *critère d'arrêt pas atteint* **faire**
5. **pour** $j = 1$ à p **faire**
6. $\vec{v}_j = \chi[\vec{v}_j + \vec{U}(0, \frac{\varphi}{2})(\vec{x}_j^* - \vec{x}_j) + \vec{U}(0, \frac{\varphi}{2})(\vec{x}_g^* - \vec{x}_j)]$
7. $\vec{x}_j = \vec{x}_j + \vec{v}_j$
8. **si** $f(\vec{x}_j) \leq f(\vec{x}_j^*)$ **alors**
9. $\vec{x}_j^* = \vec{x}_j$
10. **si** $f(\vec{x}_j^*) \leq f(\vec{x}_g^*)$ **alors**
11. $\vec{x}_g^* = \vec{x}_j^*$
12. **fin si**
13. **fin si**
14. **fin pour**
15. **fin tant que**

Dans le cadre de notre étude, il a été choisi d'utiliser la forme avec facteur de constriction de l'algorithme. En effet, Kennedy (2003), un des co-inventeurs de l'algorithme considère cette forme de l'algorithme comme étant la forme canonique de l'algorithme. Après quelques essais préliminaires, l'utilisation du facteur de constriction s'est avérée beaucoup plus souple et universelle que la méthode V_{max} arbitraire. Aussi, il a été choisi d'utiliser 40 particules.

3.1.8 Escalade stochastique par distributions gaussiennes (SHCLVND)

L'algorithme SHCLVND (*Stochastic Hill Climber with Learning by Vectors of Normal Distributions*) Rudolf & Köppen (1996) est très simple et utilise des mécanismes rappelant ceux de la stratégie d'évolution. Ses fondements reposent strictement sur le contrôle de distributions de probabilités pour la génération de candidats améliorés. La recherche se fait en ajustant les moyennes et les écart-types de distributions gaussiennes constituant le vecteur de probabilités $\vec{P}$ (équation).

$$\vec{P} = \vec{N}(\vec{\mu},\vec{\sigma}) = \left\{\vec{N}_1(\vec{\mu}_1,\vec{\sigma}_1),\vec{N}_2(\vec{\mu}_2,\vec{\sigma}_2),....,\vec{N}_d(\vec{\mu}_d,\vec{\sigma}_d)\right\} \tag{2.22}$$

L'initialisation de $\vec{P}$ se fait à l'aide des équations (2.23), (2.24) et (2.25) :

$$\vec{r}_{range} = \vec{h} - \vec{l} \tag{2.23}$$

$$\vec{\mu} = \vec{l} + \frac{\vec{r}_{range}}{2} \tag{2.24}$$

$$\vec{\sigma} = \vec{r}_{range} \cdot \Psi \tag{2.25}$$

Ici, Ψ est la constante d'étalement initial qui doit être choisie de façon à ce que la totalité du domaine de recherche puisse être initialement couvert avec une probabilité suffisante. Rudolf & Köppen (1996) proposent d'utiliser $\Psi = 0.5$. L'algorithme resserre ensuite les écart-types des distributions normales itérativement et linéairement de manière à passer d'une recherche globale permettant d'explorer tout l'hyperespace, vers une recherche plus locale qui permet

de tendre vers l'optimum global en concentrant les recherches autour de la moyenne. Contrairement à l'algorithme ES, pour qui l'adaptation des écart-types est définie selon la règle du 1/5, dans le cas présent, l'adaptation des écart-types est prédéterminée et strictement décroissante. Cela signifie que l'algorithme finit nécessairement par s'écraser sur lui-même mettant ainsi fin à l'évolution à cause de la granularité finie des systèmes informatiques. Le vecteur moyen $\vec{\mu}$ se repositionne à chaque génération là où les solutions explorées semblent les meilleures. Soit $\vec{x}_{moy}$, la moyenne des Υ meilleurs individus parmi la population de p individus d'une génération obtenue à l'aide de l'équation (2.26). L'archive des Υ meilleurs individus se nomme l'archive B.

$$\vec{x}_{moy} = \frac{\sum_{j=1}^{\Upsilon} \vec{x}_j}{\Upsilon} \quad \Upsilon \leq p \tag{2.26}$$

L'évolution du vecteur de probabilité $\vec{P}$ est faite de la manière suivante :

$$\vec{\mu} = \vec{\mu} + \Delta\left(\vec{x}_{moy} - \vec{\mu}\right) \tag{2.27}$$

$$\vec{\sigma} = \Gamma \cdot \vec{\sigma} \tag{2.28}$$

Les auteurs proposent également que la valeur finale de $\vec{\sigma}$ soit mille fois plus petite que sa valeur initiale. Suivant cette logique :

$$\Gamma = \left(\frac{1}{1000}\right)^{\frac{1}{N_{max}}} \tag{2.29}$$

3.1.8.1 Notre implémentation

Rudolf & Köppen (1996) propose que $\Upsilon = 3$, $p = 200$, $\Delta = 0.5$ et $\Gamma = (1/1000)^{(1/2500)}$. Cela signifie également qu'ils limitaient le nombre d'itérations à 2500. Il s'est avéré dans notre cas, qu'avec ce paramétrage, l'algorithme n'avait généralement pas terminé son évolution après les 2500 itérations allouées. Nous avons donc choisi de conserver ces valeurs canoniques, mais de dissocier la valeur de Γ du nombre d'itérations à effectuer.

Algorithme 5 SHCLVND

Arguments :

Ψ : Constante d'étalement initial
Δ : Coefficient d'évolution
Υ : Nombre d'individus retenu pour la sélection
B : Archive des Υ meilleurs individus
Γ : Coefficient de réduction
p : Nombre d'individus

1: $\vec{r}_{range} = \vec{h} - \vec{l}$
2: $\vec{\mu} = \vec{l} + \frac{\vec{r}_{range}}{2}$
3: $\vec{\sigma} = \vec{r}_{range} \cdot \Psi$
4: **tant que** *critère d'arrêt pas atteint* **faire**
5: $\quad \vec{B} = \emptyset$
6: $\quad$ **pour** $i = 1$ à p **faire**
7: $\quad\quad \vec{x} = \vec{P}$
8: $\quad\quad$ **si** $|\vec{B}| < \Upsilon$ **alors**
9: $\quad\quad\quad \vec{B} = \vec{B} \cup \{\vec{x}\}$
10: $\quad\quad$ **sinon si** $f(\vec{x}) < f(\vec{x}^*)$ **alors**
11: $\quad\quad\quad \vec{B} = \left(\vec{B} \setminus \{\vec{x}^*\}\right) \cup \vec{x}$
12: $\quad\quad\quad$ *Réajuster* $\vec{x}^*$
13: $\quad\quad$ **fin si**
14: $\quad$ **fin pour**
15: $\quad \vec{x}_{moy} = \sum_{\vec{x} \in \vec{B}} \frac{\vec{x}}{\Upsilon}$
16: $\quad \vec{P} = N(\vec{\mu} + \Delta(\vec{x}_{moy} - \vec{\mu}), \vec{\sigma} \cdot \Gamma)$
17: **fin tant que**

CHAPITRE 4

DÉFINITION DU CADRE EXPÉRIMENTAL – LE BANC DE TEST

La section 1.3.3 a permis d'exposer le lien qui existe entre les performances des algorithmes d'optimisation et le problème à optimiser. Le principe de classe de problèmes et sa portée ont également été soulevés. Le résultat d'une comparaison des performances de deux algorithmes différents sur un problème donné doit être analysé sous certaines réserves, car les performances sont directement liées au problème en question. Il est toutefois généralement accepté d'augmenter l'étendu des résultats sur les problèmes d'une même classe. La construction d'un banc de test doit tenir compte du principe de classe des problèmes. La portée d'une étude dépend directement des problèmes du banc de test sur lequel les algorithmes son comparés. Dans le cas présent, l'objectif est de couvrir du mieux possible plusieurs classes de problèmes. La classification même des problèmes, du point de vue des métaheuristiques, est complexe et les caractéristiques qui déterminent la classe d'un problème sont relativement mal connues. Toutefois, bien que le lien entre les caractéristiques d'un problème et son appartenance à une classe soit difficile à établir, un certain nombre de caractéristiques des problèmes peuvent être soulevées et quantifiées et puisque toutes les métaheuristiques suivent un même principe directeur, il est possible d'établir un lien entre ces caractéristiques et le niveau de difficulté relatif du problème.

4.1 Les caractéristiques d'un problème

4.1.1 La modalité

Une fonction est dite unimodale lorsqu'elle ne contient seulement qu'un seul optimum. Une fonction unimodale est une fonction entièrement convexe sur tout son domaine de recherche. Les fonctions unimodales sont généralement très simples à optimiser puisqu'elles ne nécessitent que des mécanismes d'optimisation locale. À l'opposé, une fonction est multimodale dès lors qu'elle contient plus qu'un minimum. La plupart du temps, un seul ou un nombre très limité de ces optimums seront globaux, les autres étant locaux. Les optimums

locaux sont des attracteurs qui piègent les algorithmes de fouille locale, rendant ainsi leurs performances médiocres. Pour un algorithme d'optimisation global, ce sont des pièges qui peuvent être déjoués mais qui constituent une complexité supplémentaire. Une augmentation de la modalité d'une fonction peut être un facteur de complexification du problème.

4.1.2 La dimensionnalité

Le nombre de dimensions d, est le nombre de variables à optimiser simultanément. Aussi, plus le nombre de variables à optimiser est élevé, plus le domaine de fouille est grand. En général, une augmentation de la dimensionnalité d'un problème emmène une augmentation du nombre d'itérations à effectuer par l'algorithme pour atteindre un même niveau d'optimisation.

4.1.3 L'épistasie entre les dimensions

Ce paramètre est un exemple éloquent de la relativité de la complexité des problèmes. L'épistasie d'un problème établit la relation existant entre les variables. Un problème peut-être à variables séparables, ou à variables inséparables. Un problème à d dimensions est séparable s'il peut être réécrit comme une sommation de d fonctions à une variable. Dans ce cas, l'épistasie entre les dimensions est nulle. Une fonction à variables inséparables implique que la modification d'une variable affecte l'implication d'une autre variable. L'épistasie peut alors être de faible à forte. L'évolution de plusieurs algorithmes fonctionne en séparant les variables et en les ajustant une à la fois. De tels algorithmes risquent d'être plus affectés par les fonctions à forte épistasie que les algorithmes qui ne séparent pas les variables durant le processus d'évolution.

4.1.4 La zone d'attraction

La zone d'attraction, ou zone de convergence, représente la zone dans laquelle un simple algorithme de fouille local basé sur le gradient peut résoudre le problème. La zone d'attraction est mesurée grossièrement et relevée sous la forme de pourcentage de

l'hyperespace de recherche total. Plus le rapport entre la zone de convergence et la fenêtre de fouille est petit, plus les algorithmes de fouille devront être précis pour atteindre l'optimum global. Évidemment, la zone d'attraction d'une fonction unimodale représente toujours 100% du domaine de recherche.

4.1.5 La régularité

Cette caractéristique ne s'applique qu'aux fonctions multimodales. La régularité de la disposition et de l'échelle des minimums locaux simplifie habituellement la recherche du minimum global. D'une génération à une autre, un schéma d'évolution similaire reste efficace et cela limite le besoin d'adaptation des algorithmes. De nombreux problèmes synthétiques sont très réguliers.

4.1.6 L'isolement de l'optimum

Cette notion ne s'applique aussi qu'aux fonctions multimodales. Elle se veut un indice du lien existant entre la position de l'optimum global et la structure générale du domaine de fouille. Par exemple, une fonction dont le niveau d'isolement est faible est une fonction dont la structure générale converge vers l'optimum et où le minimum global est entouré de minimums locaux dont la performance est proportionnelle à la proximité du minimum global. À l'inverse, une fonction dont l'optimum est à l'opposé de ce vers quoi la structure générale et les minimums locaux guident l'algorithme possède un haut niveau d'isolement. Le niveau d'isolement est un bon indicateur de la valeur de l'information pouvant être extraite de la structure du problème. Plus le niveau d'isolement est élevé, plus il sera difficile pour une métaheuristique d'atteindre l'optimum.

4.1.7 La grandeur du domaine de fouille

La grandeur du domaine de fouille est un facteur qui a pour effet de ralentir ou accélérer la progression des algorithmes vers l'optimum. Il affecte également la zone de convergence. Agrandir le domaine de fouille augmente le nombre de solutions possibles, réduit la zone de

convergence, peut engendrer une augmentation de la modalité et lorsque la régularité d'une fonction est faible, cela peut influencer l'isolement de l'optimum global.

4.2 Les fonctions de test

4.2.1 Ajustement des fonctions

Toutes les fonctions du banc de test sont des fonctions connues et largement utilisées pour l'évaluation des performances d'algorithmes d'optimisations. Par un souci d'uniformité et dans le but d'alléger l'analyse des résultats et de simplifier la compréhension et l'implémentation des fonctions de comparaisons, toutes les fonctions de tests dont la valeur de coût de l'optimum global n'est pas originalement de zéro, ont été ajustées à l'aide d'une constante ζ supplémentaire de manière à répondre à cette particularité. Aussi, l'optimum global de plusieurs fonctions est centré sur le domaine de fouille suggéré. Cela introduit nécessairement un biais surtout pour l'algorithme SHCLVND pour qui la population initiale suit une distribution normale centrée par rapport au domaine de recherche. Pour ces fonctions, un vecteur de translation du domaine de fouille a été ajouté de manière à décentrer l'optimum global. Le choix du vecteur de translation $\overline{tr}$ du domaine de chaque fonction a été fait de manière purement aléatoire à l'aide de l'équation(2.30). Les mêmes valeurs on été utilisées pour tous les tests et sont disponibles à l'Annexe III.

$$tr_i = \frac{(1-\alpha)(|h_i - l_i|)}{2} + \alpha(|h_i - l_i|)U(0,1) \quad \forall i \in \{1,2,....,d\} \tag{2.30}$$

De plus, pour différentes raisons expliquées lors de la description de chaque fonction, certaines modifications additionnelles furent apportées à certains problèmes, dans le but de préserver la rigueur de l'analyse. L'annexe IV contient les caractéristiques de chaque problème.

4.2.1.1 Fonction de DeJong #1 (sphere)

$$f_1(\vec{x}) = \sum_{j=1}^{d} x_j^2 \tag{2.31}$$

$$\vec{x}^* = \{\vec{0}\}$$

$$\vec{tr} + (\overrightarrow{-5.12}) \leq \vec{x} \leq 5.12 + \vec{tr}$$

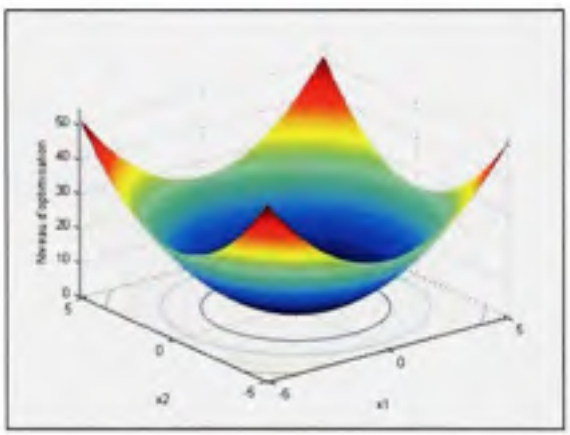

Figure 4.1 ***Fonction de DeJong #1 (sphere) sous la forme 2D.***

Cette fonction est une des cinq fonctions utilisées en 1975 par DeJong (1975). Cette fonction se retrouve classiquement dans la plupart des bancs de test. C'est un simple paraboloïde elliptique dont l'optimum se situe à l'origine. C'est donc une fonction unimodale (convexe), continue et quadratique. Il n'y a aucun effet d'épistasie entre les dimensions puisque le problème est à variable séparable. Comme la fonction est unimodale, tout le domaine de fouille converge vers le seul optimum de la fonction et la zone d'attraction couvre tout le domaine. La fonction est parfaitement symétrique autour de l'optimum. Cette fonction est très simple à optimiser et n'importe quel algorithme sérieux d'optimisation doit être en mesure d'atteindre facilement son optimum. Les variations de performances se situent plutôt au niveau de la vitesse de convergence que de la capacité de convergence. Le domaine de fouille a été déplacé pour décentrer l'optimum.

4.2.1.2 Fonction hyper-ellipsoïde

$$f_2(\vec{x}) = \sum_{j=1}^{d} j \cdot x_j^2 \tag{2.32}$$

$$\vec{x}^* = \{\vec{0}\}$$

$$\overrightarrow{tr} + (\overrightarrow{-5.12}) \leq \vec{x} \leq 5.12 + \overrightarrow{tr}$$

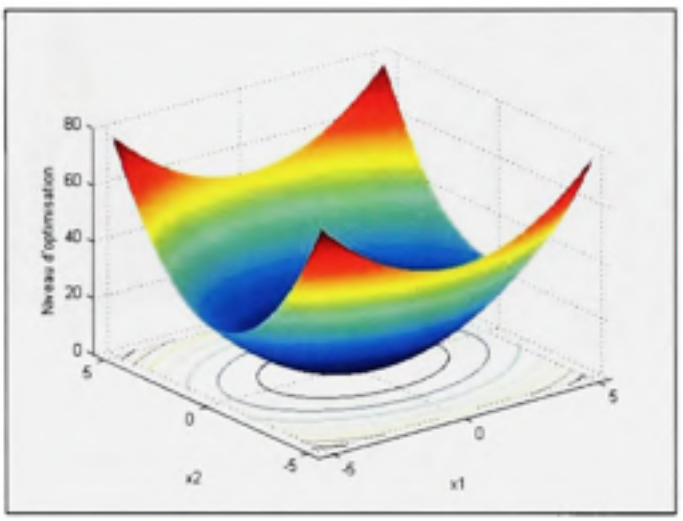

Figure 4.2 ***Fonction hyper-ellipsoïde sous la forme 2D.***

Cette fonction est une variante de la fonction DeJong #1 précédente. La seule différence provient de l'ajout d'un coefficient d'implication de chaque dimension. Ces coefficients limitent la symétrie du problème. Théoriquement, les performances des algorithmes ne devraient pas vraiment être affectées par cette modification et dans le cas contraire, cela signifierait une faiblesse de l'algorithme.

4.2.1.3 Fonction Rosenbrock's saddle

$$f_3(x_1, x_2) = 100 \cdot \left(x_1^2 - x_2\right)^2 + (1 - x_1)^2 \tag{2.33}$$

$$\vec{x}^* = \{1,1\}$$

$$\overline{-2.048} \le \vec{x} \le \overline{2.048}$$

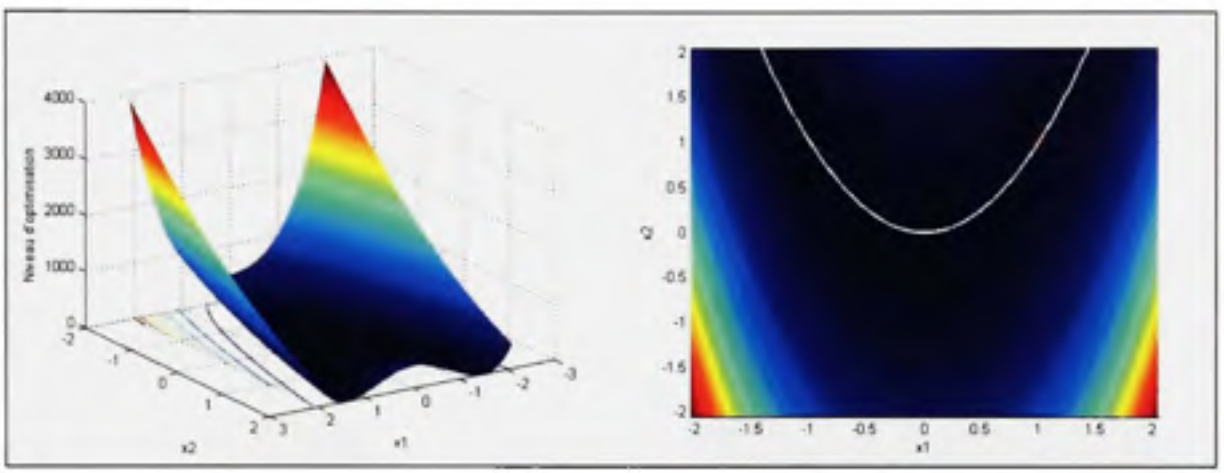

Figure 4.3 ***Fonction Rosenbrock's saddle.***

Cette fonction, également issue de DeJong (1975) est quartique unimodale, non-convexe et continue à deux dimensions inséparables. Le minimum se situe au coeur d'une vallée parabolique suivant la courbe $x_2 = x_1^2$ (tracé jaune sur l'image de droite de la figure(3.4)). En fait, un algorithme d'optimisation local pourra se rendre rapidement à cette vallée mais il devient ensuite plus difficile de converger vers le minimum global en suivant cette vallée non-linéaire créée par l'épistasie entre les deux dimensions. Toutefois, étant une fonction unimodale, les algorithmes d'optimisation globale n'ont généralement aucune difficulté à optimiser cette fonction. L'épistasie aura toutefois tendance à ralentir la convergence.

4.2.1.4 Fonction Rcos de Branin

$$f(x_1,x_2)=a\left(x_2-bx_1^2+cx_1-d\right)^2+e(1-f)\cos(x_1)+e+\zeta \tag{2.34}$$

$$a=1 \quad b=\frac{5}{4\pi^2} \quad c=\frac{5}{\pi} \quad d=6 \quad e=10$$

$$\vec{x}^* = \{-\pi,12.25\} \wedge \{\pi,2.25\} \wedge \{3\pi,2.25\}$$

$$\vec{L} \le \vec{x} \le \vec{U}$$

$$\vec{L} = \{-5,0\}$$

$$\vec{U} = \{10,15\}$$

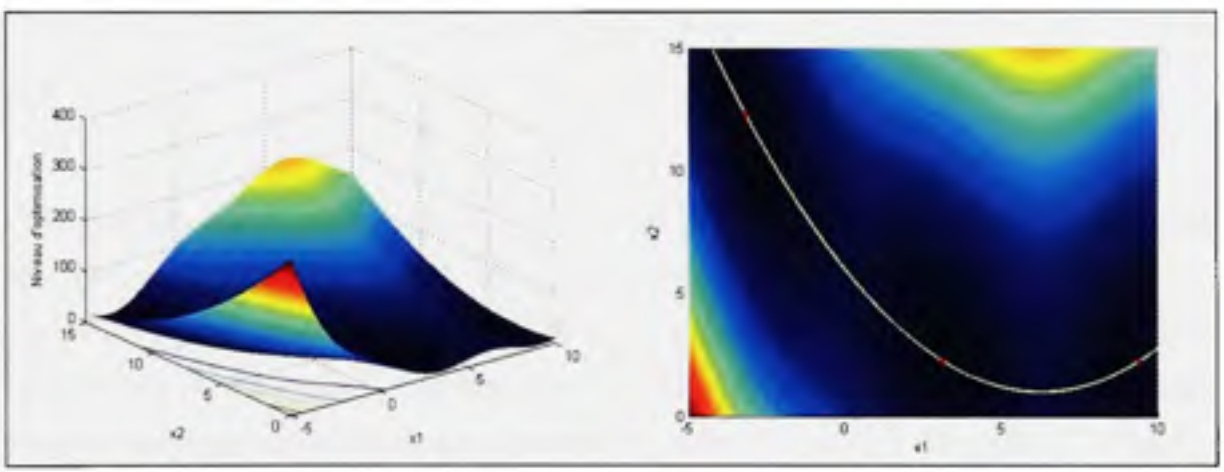

Figure 4.4 ***Fonction rcos de Branin (gauche), avec la trajectoire parabolique et les trois optimums (droite).***

Cette fonction Branin (1972) continue à variables inséparables est plutôt particulière dans notre banc de test puisqu'elle possède trois minimums qui sont tous équivalents, donc globaux. La fonction est ainsi multimodale mais comme tous les minimums sont globaux, la modalité est ici une simplification pour les algorithmes de fouilles puisqu'ils peuvent converger avec succès vers l'un ou l'autre des minimums. Cette particularité fait en sorte que la zone d'attraction couvre 100% du domaine même si la fonction est multimodale. Le problème est proposé ici dans une version en deux dimensions uniquement et l'épistasie est importante. En ressemblance à la fonction de Rosenbrock, la fonction possède une vallée qui

suit une courbe parabolique d'équation $x_2 - b \cdot x_1^2 + c \cdot x_1 - d = 0$. Par contre, cette fois la vallée est ondulée par le terme en cosinus et ce sont ces ondulations qui créent les trois minimums. La répartition des minimums n'est pas régulière. Les minimums ne sont pas isolés puisqu'ils sont positionnés sur la vallée. En somme, même si la fonction est multimodale, comme chaque minimum est une solution équivalente, une stratégie de fouille locale est suffisante et les AEs doivent réussir sans difficulté à optimiser cette fonction.

4.2.1.5 Fonction six-humps camel

$$f_5(x_1, x_2) = \left(4 - 2.1x_1^2 + \frac{x_1^4}{3}\right)x_1^2 + x_1 x_2 + \left(-4 + 4x_2^2\right)x_2^2 + \zeta \tag{2.35}$$

$$\vec{x}^* = \{-0.0898, 0.7126\} \wedge \{0.0898, -0.7126\}$$

$$\overline{-10} \leq \vec{x} \leq \overline{10}$$

Cette fonction multimodale continue, Lee & Geem (2004), contient six minimums dont deux sont globaux. La relation entre les deux dimensions de ce problème (l'épistasie) est légère. Vus de l'ensemble du domaine, les minimums globaux sont positionnés au centre d'un creux couvrant une bonne partie de la surface totale, l'isolement des minimums est ainsi faible. Le domaine est composé de deux zones symétriques créant une certaine régularité. Un procédé de fouille uniquement basé sur le gradient risque fortement de figer dans l'un des quatre minimums locaux. À l'opposé, un algorithme qui comporte des mécanismes de recherche globale ne devrait pas éprouver de problème à optimiser la fonction et devrait même pouvoir tirer profit des minimums locaux pour s'approcher de l'une des deux solutions optimales dont la zone d'attraction (pour les deux) couvre environ 1% du domaine de fouille.

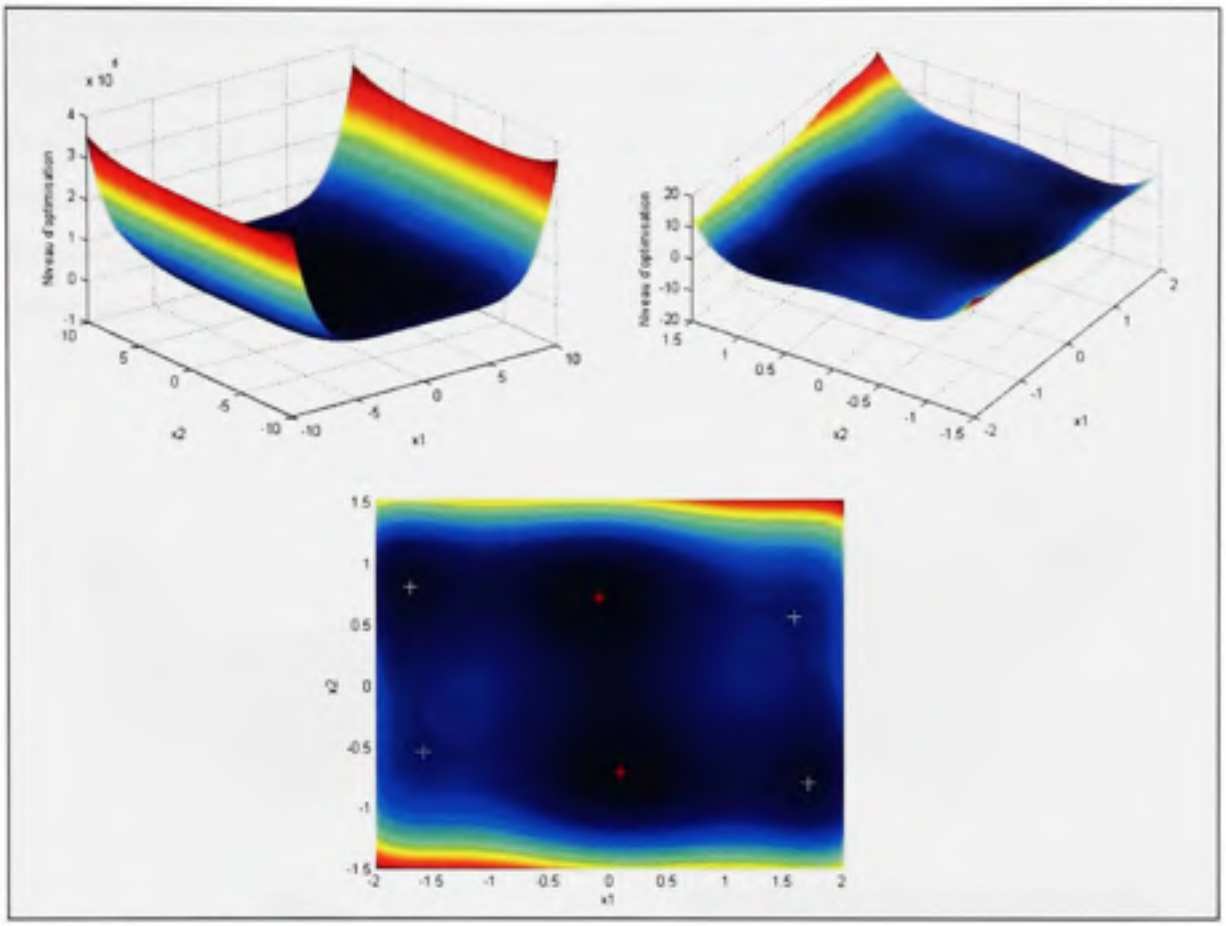

Figure 4.5 ***Fonction "six-hump camel" (haut-gauche), avec réduction de la fenêtre (haut-droite) et les six minimums (bas).***

4.2.1.6 Fonction Goldstein-Price

$$f_6(x_1,x_2)=\left[1+(x_1+x_2+1)^2\cdot\left(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2\right)\right]\cdot \left[30+(2x_1-3x_2)^2\cdot\left(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2\right)\right]+\zeta \tag{2.36}$$

$$\vec{x}^* = \{0,-1\}$$

$$\overline{-2} \le \vec{x} \le \vec{2}$$

Cette fonction Goldstein & Price (1971) est un problème continu à deux dimensions inséparables. À l'intérieur de la fenêtre de recherche, on dénombre quatre minimums dont un seul est global. En observant différentes coupes de la fonction, on peut remarquer que les

quatre minimums sont situés sur différents paliers d'une vallée bien définie mais dont l'échelle est minime en rapport aux sommets de la fonction. Il n'existe aucune régularité dans la disposition des minimums et l'épistasie entre les deux dimensions est responsable des trajectoires linéaires angulées de la vallée. Les trois minimums locaux sont sur la trajectoire du minimum global, évitant ainsi qu'il ne soit isolé mais la zone de convergence reste petite (environ 0.41%).

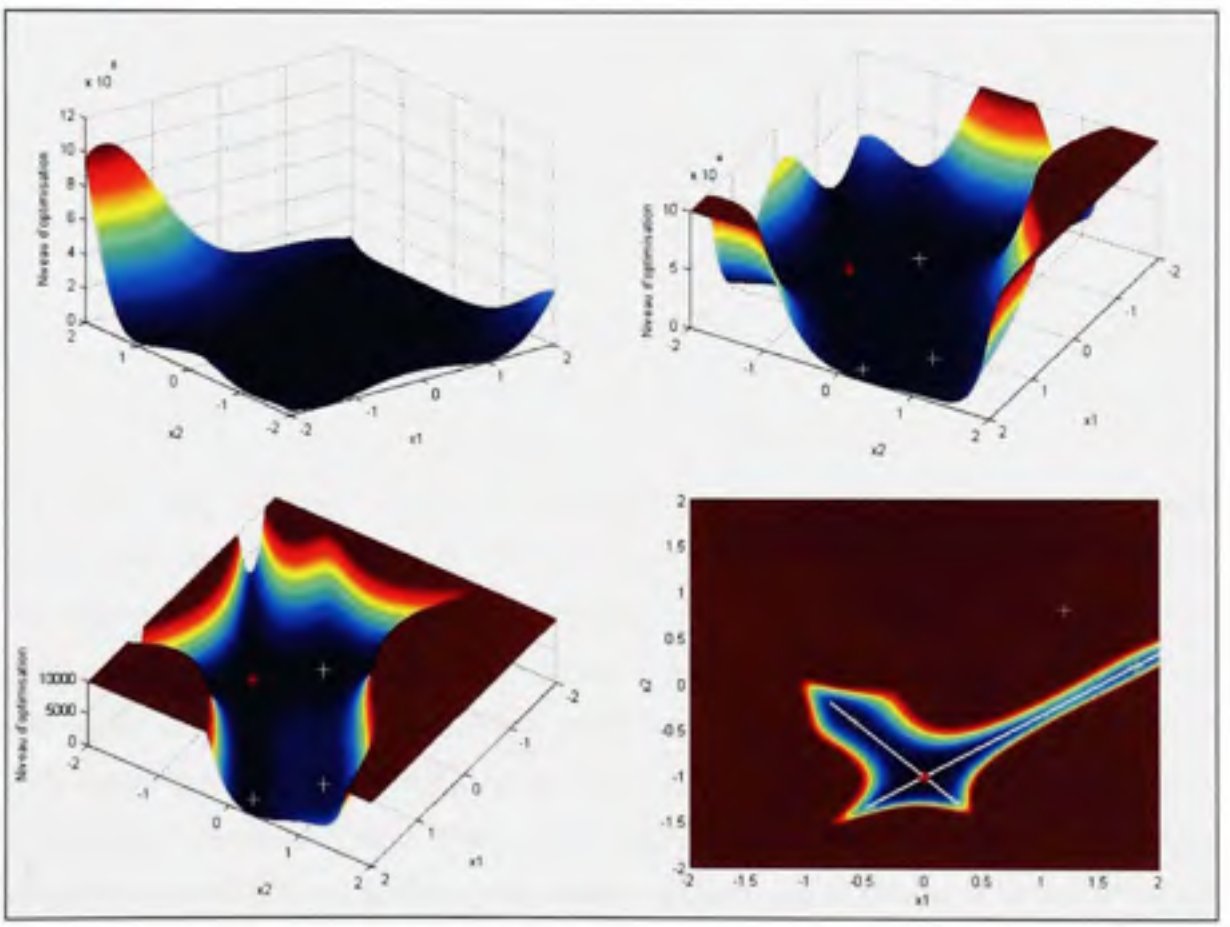

Figure 4.6 ***Fonction de Goldstein-Price (haut-gauche), avec les minimums et une coupe à 100 000 (haut-droite), une coupe à 10 000 (bas-gauche) et une coupe à 350 avec les deux trajectoires linéaires (bas- droite).***

4.2.1.7 Fonction Shekel's Foxholes

$$f_7(x_1,x_2)=\frac{1}{0.002+\sum_{j=1}^{25}\frac{1}{j+\sum_{i=1}^{2}(x_i-a_{i,j})^6}}+\zeta \qquad (2.37)$$

$$\begin{aligned}
\{a_{1,1},\ a_{1,6},\ a_{1,11},\ a_{1,16},\ a_{1,21},\ a_{2,1},\ a_{2,2},\ a_{2,3},\ a_{2,4},\ a_{2,5}\} &= -32\\
\{a_{1,2},\ a_{1,7},\ a_{1,12},\ a_{1,17},\ a_{1,22},\ a_{2,6},\ a_{2,7},\ a_{2,8},\ a_{2,9},\ a_{2,10}\} &= -16\\
\{a_{1,3},\ a_{1,8},\ a_{1,13},\ a_{1,18},\ a_{1,23},\ a_{2,11},\ a_{2,12},\ a_{2,13},\ a_{2,14},\ a_{2,15}\} &= 0\\
\{a_{1,4},\ a_{1,9},\ a_{1,14},\ a_{1,19},\ a_{1,24},\ a_{2,16},\ a_{2,17},\ a_{2,18},\ a_{2,19},\ a_{2,20}\} &= 16\\
\{a_{1,5},\ a_{1,10},\ a_{1,15},\ a_{1,20},\ a_{1,25},\ a_{2,21},\ a_{2,22},\ a_{2,23},\ a_{2,24},\ a_{2,25}\} &= 32
\end{aligned}$$

$$\vec{x}^{*}=\{-32,-32\}$$

$$\overline{-65.536}\le\vec{x}\le\overline{65.536}$$

Cette fonction, issue de la suite de DeJong (DeJong (1975)), constitue un problème multimodal de difficulté modérée. Cette fonction non-convexe continue à deux variables non-séparables comporte 25 minimums sous la forme de cavités régulières, positionnées suivant un quadrillé parallèle aux axes des deux variables, faisant en sorte que l'épistasie est très faible. Les aspérités sont creusées à même une surface qui semble parfaitement plane mais qui est pourtant un prolongement en tous points des 25 aspérités. La disposition de la profondeur maximale des trous suit approximativement un plan d'équation $-0.00544\cdot x_1-0.0233\cdot x_2+0.0806\cdot z=1$ faisant en sorte que le plus profond des trous est positionné en $(-32,-32)$ et que moins profond est situé en (32, 32). La totalité du domaine est pour ainsi dire un pseudo-plan percé orienté vers la plus profonde aspérité. La zone de convergence pure de la fonction correspond approximativement à l'intervalle $l_i\le x_i\le -24 \quad i=1,2$. L'isolement du minimum global est faible puisque la profondeur des cavités s'accentue en s'approchant de l'optimum global, dirigeant ainsi la recherche dans la bonne direction et aussi parce que le pseudo-plan est aussi incliné vers

l'optimum. La difficulté de cette fonction est de se sortir des minimums locaux qui sont très profonds, abruptes et distribués sur la trajectoire d'optimisation de la fonction. Un algorithme qui ne contient pas de stratégie globale restera prisonnier de la première cavité rencontrée.

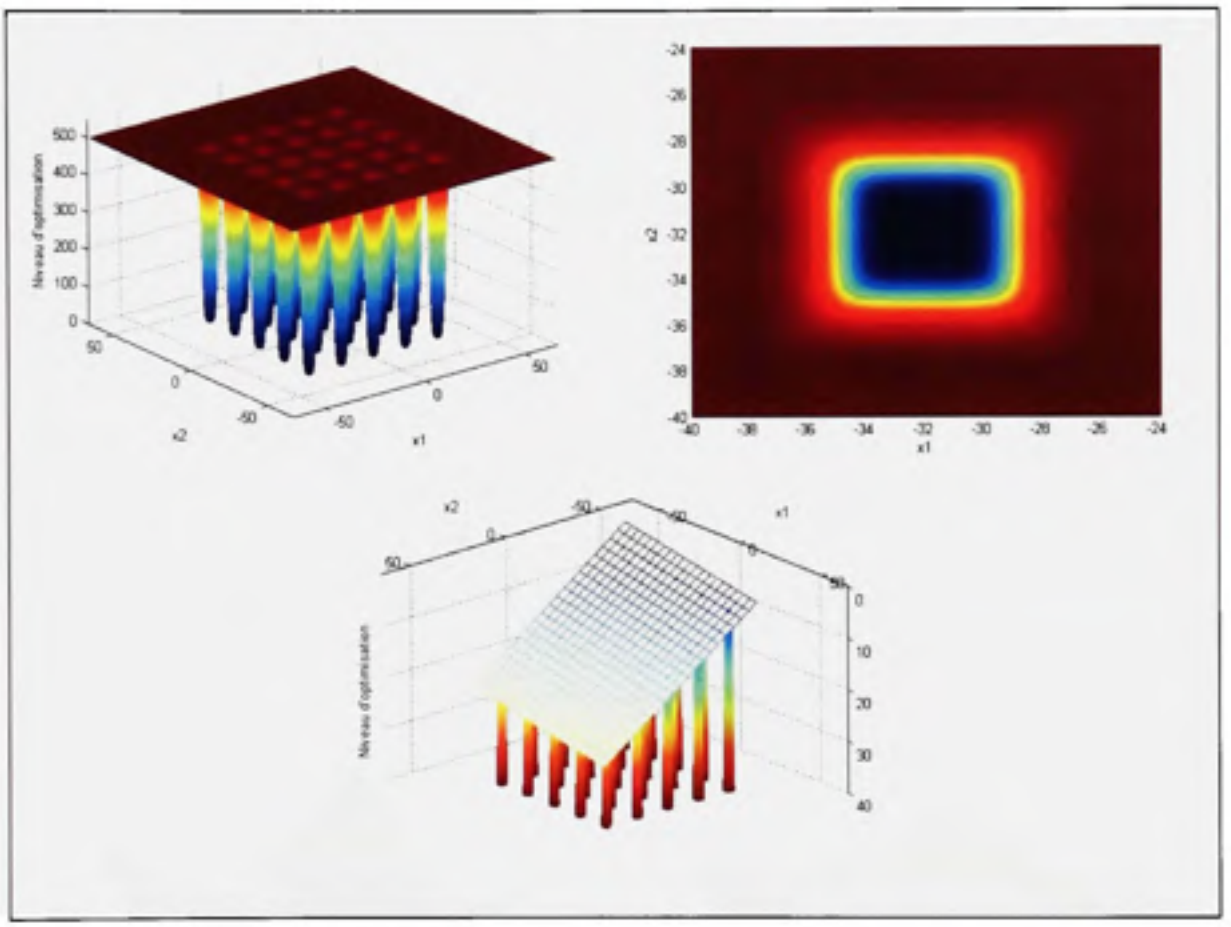

Figure 4.7 ***Fonction #5 de De Jong (Shekel's foxholes) (haut-gauche), vue de dessus de la cavité contenant l'optimum (haut-droite) et le plan des profondeurs des cavités avec une coupe à 40 (bas).***

4.2.1.8 Fonction de Rastrigin généralisée

$$f_8(\vec{x}) = 10d + \sum_{j=1}^{d}\left(x_j^2 - 10\cos\left(2\pi \cdot x_j\right)\right) + \zeta \tag{2.38}$$

$$\vec{x}^* = \left\{\vec{0}\right\}$$

$$\overrightarrow{tr} + \left(\overrightarrow{-600}\right) \le \vec{x} \le \overrightarrow{600} + \overrightarrow{tr}$$

La fonction de Rastrigin est une fonction continue à variables séparables. En réalité, cette fonction est la fonction de DeJong #1, de dimensionnalité variée, à laquelle est ajouté un modulateur de la forme $\cos\left(2\pi \cdot x_j\right)$ $i = 1,2,...,d$. Ce modulateur rend la fonction multimodale en ajoutant de nombreux minimums locaux disposés à intervalles réguliers. L'ajout du modulateur n'affecte pas la séparabilité des variables conservant ainsi l'épistasie nulle de la fonction sphérique. La zone d'attraction pure est extrêmement restreinte, puisqu'elle ne couvre que la fenêtre $-0.5 \le x_i \le 0.5$ $i = 1,2,...,d$.

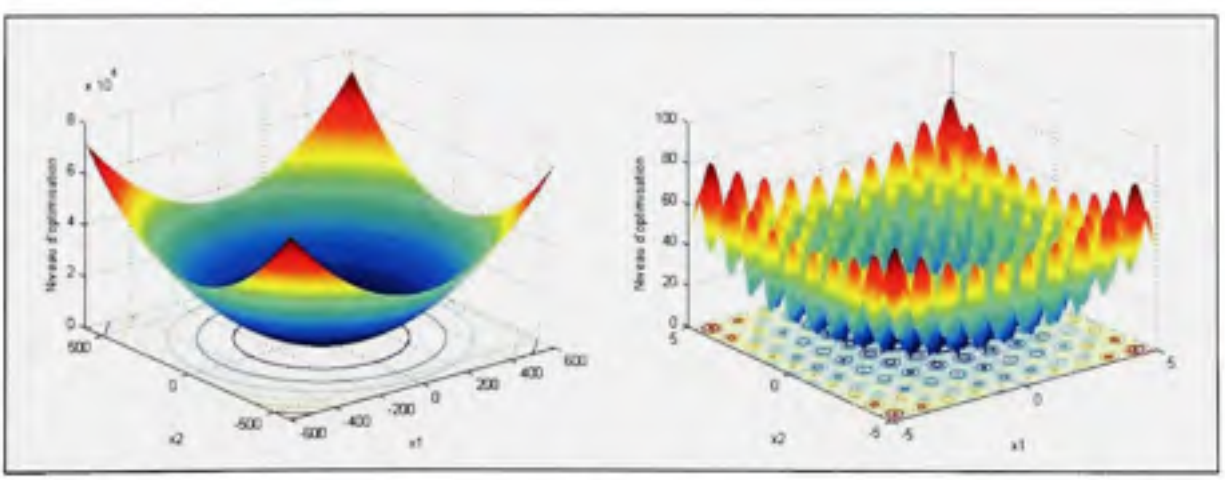

Figure 4.8 ***Fonction de Rastrigin en 2D entière (gauche), avec un rétrécissement de la fenêtre (droite).***

La fonction globale sphérique sur laquelle sont superposées les ondulations est toutefois un excellent guide vers l'optimum le rendant non-isolé. L'optimisation de cette fonction

nécessite des algorithmes de fouille, une exploration suffisamment large pour outrepasser les ondulations régulières. Par contre, la régularité des minimums et la séparabilité des deux dimensions peuvent simplifier le problème. Notons également que cette fonction est un bon exemple démontrant que l'augmentation de la modalité d'une fonction augmente sa complexité. Originalement, cette fonction n'existait que sous sa forme à deux dimensions. L'augmentation de la dimensionnalité augmente la complexité de ce problème, dont les bornes originales ont été déplacées, en réduisant le ratio de la zone de convergence.

4.2.1.9 Fonction de Ackley

$$f_9(\vec{x}) = -20 \cdot \exp\left(-0.2\sqrt{\frac{1}{d}\sum_{j=1}^{d} x_j^2}\right) - \exp\left(\frac{1}{d}\sum_{j=1}^{d}\cos(2\pi \cdot x_j)\right) + 20 + \exp(1) \qquad (2.39)$$

$$\vec{x}^* = \{\vec{0}\}$$

$$\overrightarrow{tr} + (\overrightarrow{-30}) \le \vec{x} \le \overrightarrow{30} + \overrightarrow{tr}$$

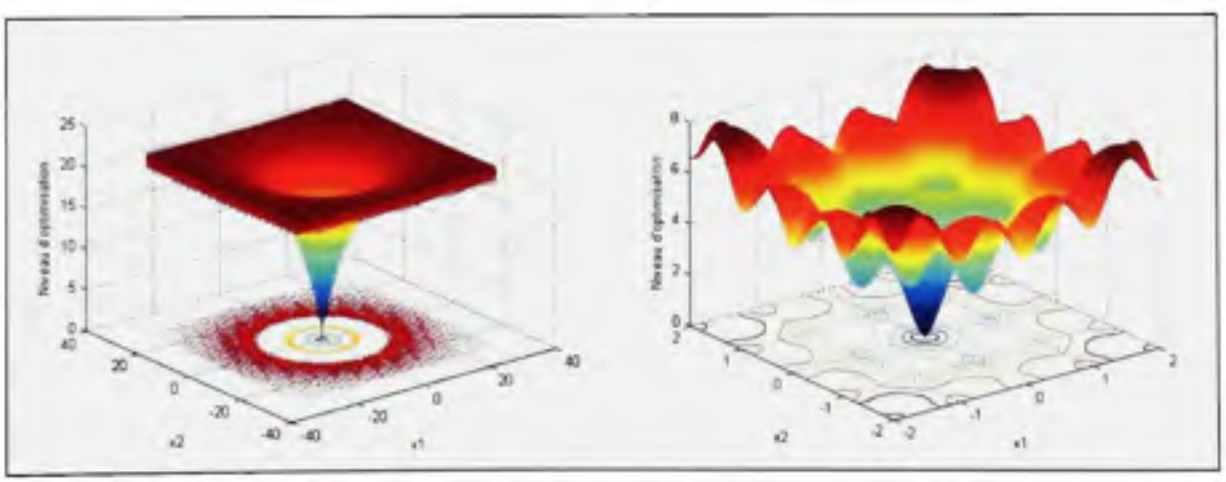

Figure 4.9 ***Fonction de Ackley en 2D entière (gauche), avec un rétrécissement de la fenêtre (droite).***

La fonction de Ackley, Ackley (1987), est une structure exponentielle à laquelle s'ajoute un modulateur dont l'effet est similaire au modulateur de la fonction de Rastrigin (exponentiel

vs sinusoïdal). Cet élément modulant génère des minimums locaux régulièrement distribués. Le niveau de difficulté de cette fonction est modéré et même si les variables sont inséparables, l'effet d'épistasie est pratiquement nul. L'isolement de l'optimum est aussi nul puisqu'il se situe au coeur de la structure exponentielle qui sert de guide. La zone d'attraction est petite, parce que l'échelle des ondulations l'est également. Pour réussir à trouver le minimum global de cette fonction, les algorithmes de fouille doivent être en mesure de dépasser les minimums locaux rencontrés en menant une fouille plus élargie. Plus l'algorithme est éloigné de l'optimum, plus il risque d'être difficile de s'extraire des attracteurs locaux puisque les informations directrices de la structure exponentielle sont moins importantes et le modulateur les camoufle.

4.2.1.10 Fonction de Shubert pénalisée

$$f_{10}(x_1, x_2) = \prod_{i=1}^{2}\left(\sum_{j=1}^{5} j\cos\left((j+1)x_i + j\right)\right) + p \tag{2.40}$$

$$p(x_1, x_2) = \beta\left[(x_1 + 1.42513)^2 + (x_2 + 0.80032)^2\right] \quad \beta = 0.5$$

$$\vec{x}^* \approx \{-1.42513, -0.80032\}$$

La fonction de Shubert à deux dimensions contient 760 minimums dont 18 sont globaux. Les variables sont inséparables et l'on ne retrouve pas vraiment d'épistasie. La fonction de Shubert ne contient originalement pas le terme p de l'équation (2.40), qui permet de modifier légèrement la fonction de manière à n'avoir qu'un seul minimum local. Le terme p Aluffi-Pentini & al.(1985), qui est la fonction sphérique à deux dimensions, est centré sur un des minimums locaux de façon à ce qu'il devienne le seul minimum global.

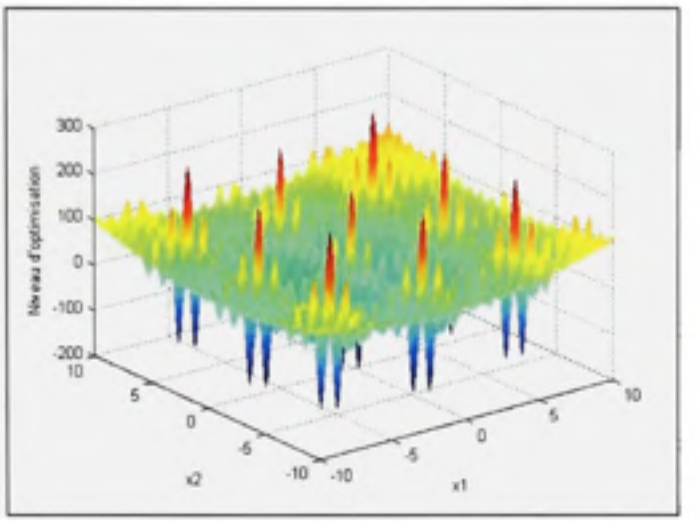

Figure 4.10 ***Fonction de Shubert pénalisée.***

Le coefficient $\beta > 0$ permet d'ajuster l'amplitude du facteur p. En choisissant $\beta = 0.5$ (petit), nous limitons l'impact de p faisant en sorte que les ondulations de la fonction originale sont beaucoup plus imposantes que la fonction sphérique superposée. La fonction est très régulière et l'isolement du minimum est faible. La zone de convergence est d'environ 0.5% du domaine de fouille.

4.2.1.11 Fonction Corana's Parabola

$$f_{11}(\vec{x}) = \sum_{j=1}^{d} \begin{cases} c_r \left(\operatorname{sgn}(z_j) + t_{j-z_j}\right)^2 d_j & \text{si } \left(|x_j - z_j| < |t_j|\right) \wedge \left(|x_j| > |t_j|\right) \\ d_j \cdot x_j^2 & \text{sinon} \end{cases} \tag{2.41}$$

$$c_r = 0.15 \quad s_j = 0.2 \quad t_j = 0.05 \quad z_j = \left\lfloor \left| \frac{x_j}{s_j} \right| + 0.4999 \right\rfloor \cdot s_j \cdot \operatorname{sgn}(x_j)$$

$$\vec{x}^* = \{\vec{0}\}$$

$$d = \begin{cases} [1\ 1000] & \text{si } n = 2 \\ \begin{bmatrix} 1 & 1000 & 10 & 10 & 1 & 10 \\ 100 & 1000 & 1 & 10 & 1 & 1000 \\ 10 & 10 & 1 & 10 & 100 & 1000 \\ 1 & 10 & 1 & 1000 & 10 & 10 \\ 1 & 10 & 100 & 1000 & 1 & 10 \end{bmatrix} & \text{si } n = 30 \end{cases}$$

Cette fonction Corana & al. (1987) discontinue et multimodale séparable (effet d'épistasie nul) est un hyper paraboloïde elliptique parsemé de cavités de profondeurs variables. Plus la distance euclidienne entre la cavité et le minimum absolu (à l'origine) est petite, moins la cavité est profonde par rapport à la surface mais plus sa valeur se rapproche de la valeur du minimum absolu. L'impact relatif des dimensions est modulé à l'aide de $d_j \quad j = 1,2,...,d$. La zone d'attraction est extrêmement réduite puisqu'elle ne couvre que la fenêtre $-0.15 \le x_j \le 0.15 \quad j = 1,2,...,d$. Par contre comme les ondulations sont sur un paraboloïde, l'isolement du minimum est nul. Le nombre de minimums équivaut à 10^{4d}. Les points de discontinuité et les profondes cavités sont les principales difficultés posées par cette fonction. Notons que t_j est responsable de la largeur des cavités, s_j, de la proximité des cavités et c_r, de la profondeur des cavités. Suivant cette logique, $t_j < (s_j/2)$ et $c_r > 0$. La fenêtre de fouille a été décalée pour décentraliser l'optimum global.

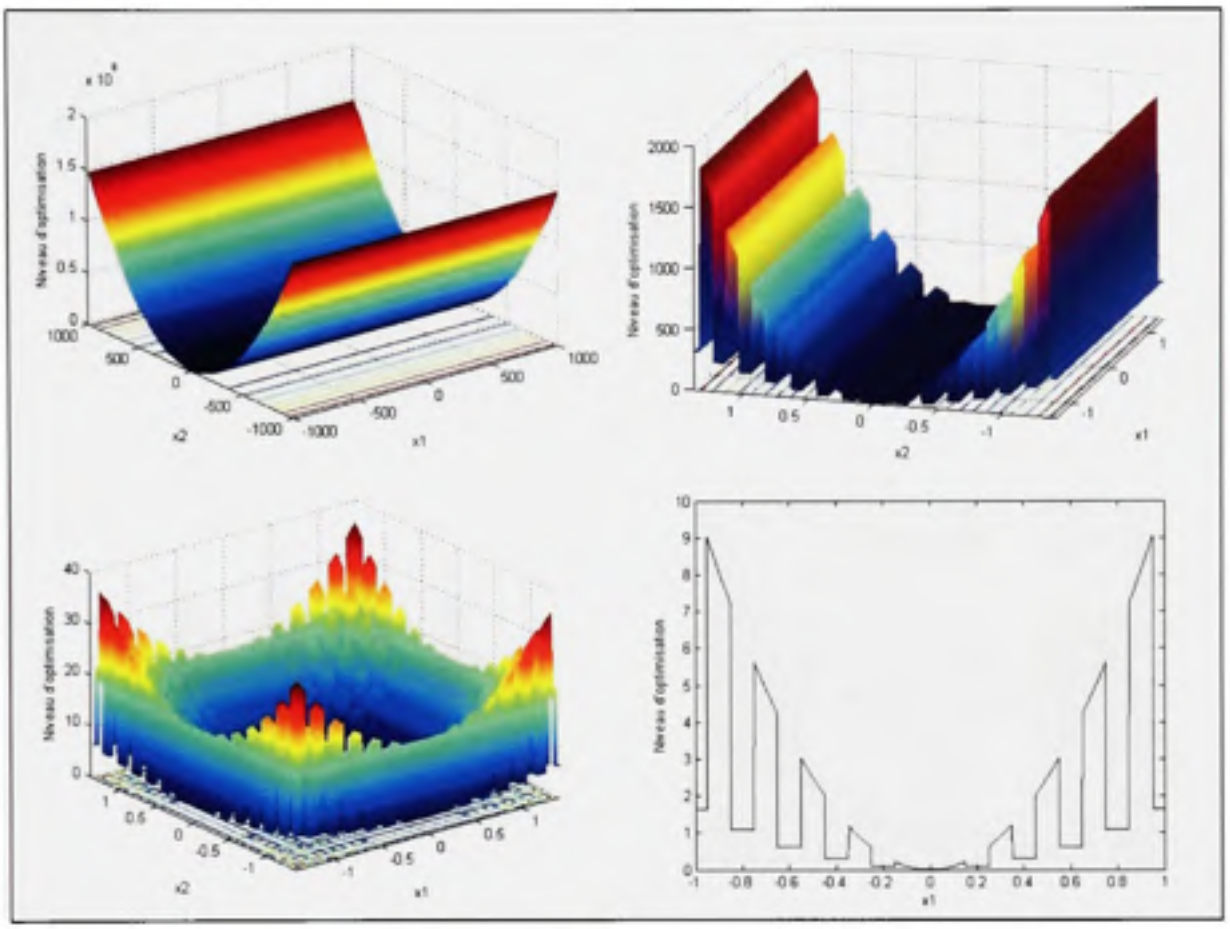

Figure 4.11 ***Fonction Corana parabola en 2D entière (haut-gauche), avec un rétrécissement de la fenêtre (haut-droite), avec d = [10, 10] (bas-gauche) et forme 1D (bas-droite).***

4.2.1.12 Fonction de Griewangk généralisée

$$f_{12}(\vec{x}) = \sum_{j=1}^{d}\left(\frac{x_j^2}{4000}\right) - \prod_{j=1}^{d}\left(\cos\left(\frac{x_j}{\sqrt{j}}\right)\right) + 1 \qquad (2.42)$$

$$\vec{x}^* = \{\vec{0}\}$$

$$\overrightarrow{tr} + \left(\overrightarrow{-600}\right) \le \vec{x} \le \overrightarrow{600} + \overrightarrow{tr}$$

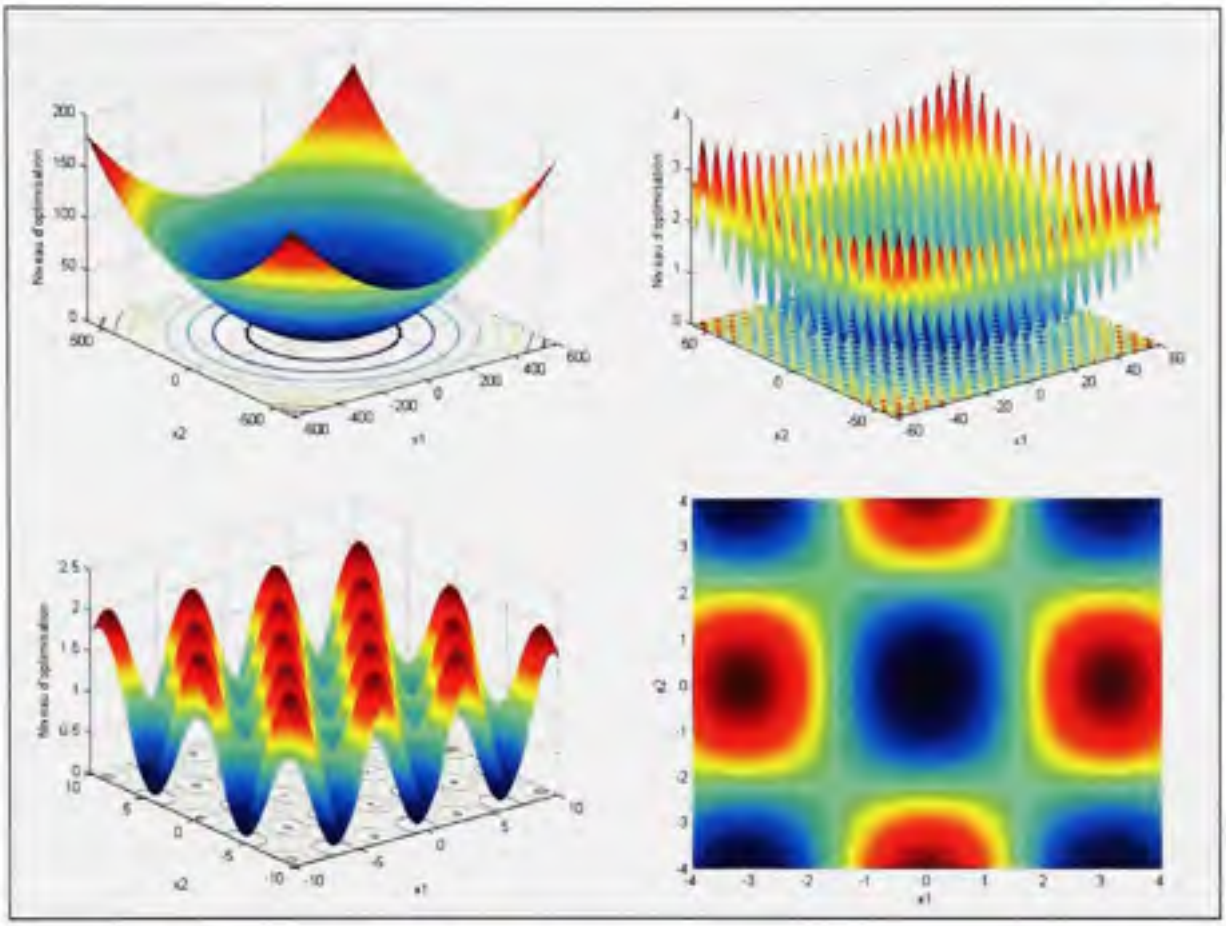

Figure 4.12 ***Fonction de Griewangk en 2D entière (haut-gauche), avec un rétrécissement de la fenêtre (haut-droite et bas-gauche) vue de dessus de la zone rapprochée du minimum global (bas-droite).***

La fonction de Griewangk est une fonction qui reprend une fois de plus la fonction sphérique DeJong #1 à laquelle un modulateur sinusoïdale multipliant les dimensions est appliqué de manière à générer des minimums locaux et à créer une relation entre les dimensions de sorte qu'elles ne sont plus séparables. Toutefois, l'effet d'épistasie reste tout de même très faible. Par contre, de simples algorithmes de fouille locale semblent très efficaces pour optimiser cette fonction lorsque le nombre de dimensions augmente. Ce paradoxe est expliqué en détail par Locatelli (2003) et confirme que la complexité diminue lorsque d augmente. Le nombre de minimums locaux régulièrement distribués augmente exponentiellement avec l'augmentation du nombre de dimensions. Les algorithmes de fouilles doivent pouvoir explorer des zones plus larges que les ondulations pour se sortir des minimums locaux et

pour être plus efficace, elles doivent être en mesure de faire évoluer efficacement toutes les dimensions simultanément.

4.2.1.13 Fonction de Schwefel modifiée

$$f_{13}(\vec{x}) = \begin{cases} \sum_{j=1}^{d} -x_j \cdot \sin\left(\sqrt{|x_j|}\right) + \zeta \quad \text{si } |x_j| \le 500 \ \forall j \in \{1,2,\ldots,d\} \\ 0 \qquad\qquad\qquad\qquad\qquad\qquad \text{sinon} \end{cases} \tag{2.43}$$

$$\vec{x}^* = \left\{\overline{420.9687}\right\}$$

$$\overline{tr} + \left(\overline{-500}\right) \le \vec{x} \le \overline{500} + \overline{tr}$$

La structure générale de cette fonction multimodale continue est très intéressante. La variation de l'amplitude de ses minimums s'amplifie d'une manière symétrique et régulière rendant cette fonction déceptive. En effet, un algorithme peut facilement se laisser guider dans la mauvaise direction s'éloignant ainsi de l'optimum. Les variables sont séparables et il n'y a aucun effet d'épistasie. Le niveau d'isolement du minimum est moyen (fonction déceptive). Aussi, la fonction fait en sorte que les minimums s'amplifient en s'éloignant de zéro de sorte que de nombreux minimums hors de la fenêtre de fouille sont meilleurs. La fonction a donc été modifiée de sorte que le niveau d'optimisation soit de zéro à l'extérieur des bornes de la fenêtre originale.

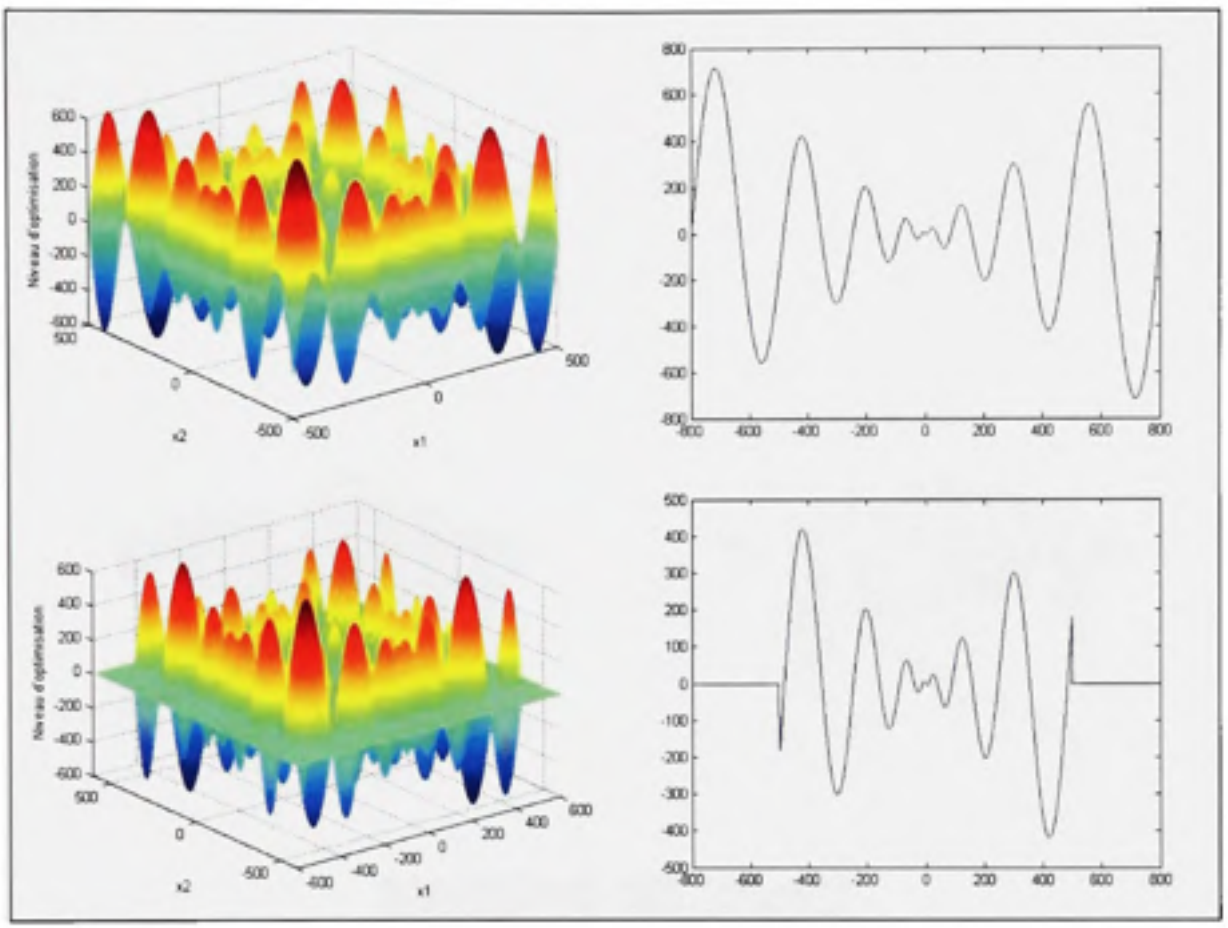

Figure 4.13 ***Fonction de Schwefel en 2D entière (haut-gauche), en 1D (haut-droite), 2D modifiée (bas-gauche) et 1D (bas-droite).***

4.2.1.14 Fonction de Michalewicz

$$f_{14}(\vec{x}) = -\sum_{j=1}^{d} \sin(x_j) \cdot \left(\sin\left(\frac{j \cdot x_j^2}{\pi} \right) \right)^{2m} \tag{2.44}$$

$$m = 10$$

$$\vec{x}^* \approx \begin{cases} [2.20291\ 1.57080] & \text{si } n=2 \\ \begin{bmatrix} 2.20291 & 1.57080 & 1.28500 & 1.92306 & 1.72047 & 1.57080 \\ 1.45441 & 1.75609 & 1.65572 & 1.57080 & 1.49773 & 1.69661 \\ 1.63008 & 1.57080 & 1.51755 & 1.66607 & 1.61633 & 1.57080 \\ 1.52891 & 1.64746 & 1.60776 & 1.57080 & 1.53627 & 1.63493 \\ 1.60190 & 1.57080 & 1.54144 & 1.62593 & 1.69765 & 1.57080 \end{bmatrix} & \text{si } n=30 \end{cases}$$

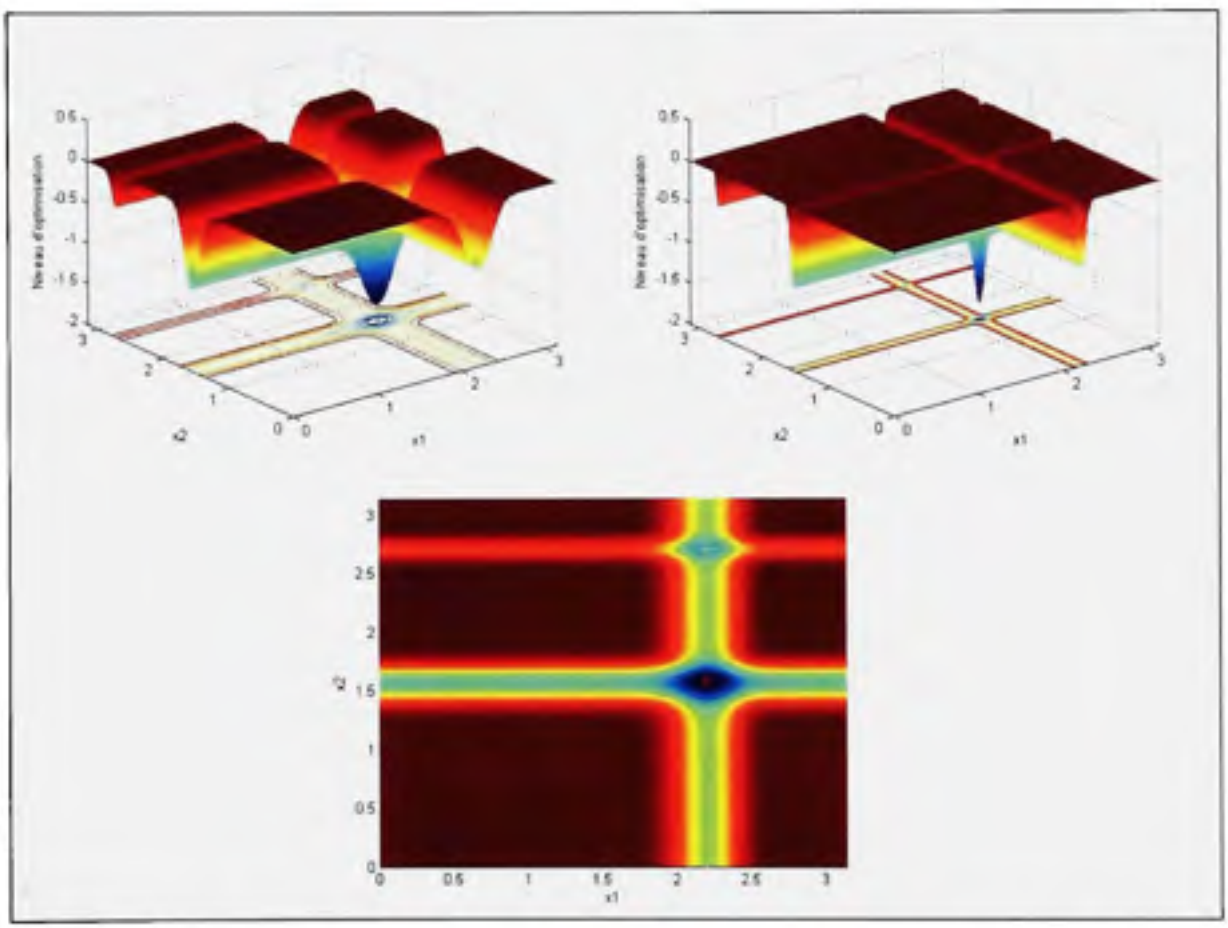

Figure 4.14 ***Fonction de Michalewicz en 2D entière (haut-gauche), avec m = 100 (haut-droite) et vue de dessus des deux minimums (bas).***

Cette fonction est sans aucun doute une des plus difficiles à optimiser du banc de tests. C'est une fonction multimodale, continue à variables séparables. Cette fonction est une surface parcourue de vallées étroites parallèles aux axes dont le nombre est directement relié au nombre de dimensions. Les minimums sont les points de rencontre des vallées et le nombre

de minimums est donc de $d!$. Même si l'épistasie est nulle, la difficulté provient de la zone de convergence qui est réduite et de l'isolement du minimum. Le minimum global est effectivement très isolé puisqu'à l'extérieure de la zone de convergence, aucune information ne permet de guider la recherche. Même la profondeur des vallées est constante. Aussi, lorsque le nombre de dimensions augmente, la zone de convergence diminue en même temps que le domaine s'élargit. Cette fonction est un véritable défi d'exploration pour les algorithmes de recherche. Ici, aucun décalage de la fenêtre de fouille originale n'est nécessaire.

CHAPITRE 5

ANALYSE DES RÉSULTATS

Notre étude avait pour principal objectif de développer un outil de comparaison statistique, permettant d'évaluer la performance des algorithmes sur différents problèmes, issus de différentes classes. L'objectif de ce chapitre n'est pas de déterminer quel algorithme fut le meilleur, puisque les algorithmes utilisés sont plutôt accessoires dans cette étude, mais plutôt d'évaluer l'étendue des informations obtenues à l'aide de la méthodologie proposée. Nous tenterons également d'évaluer si notre méthodologie nous permet d'arriver à des résultats foncièrement différents de ceux que nous aurions obtenus d'une analyse conventionnelle et d'analyser les raisons de ces différences. Il est aussi important de noter qu'il est évident qu'il faut un nombre d'itérations minimum pour voir apparaître des différences entre les algorithmes, et que les premières itérations ne peuvent être considérées dans l'étude puisqu'elle constitue en quelques sorte une période transitoire d'ajustement des algorithmes. Toutefois, dans le présent travail de recherche, cette phase transitoire ne correspond jamais à plus de 10% du domaine temporel complet de l'étude. Principalement, la méthodologie proposée devait initialement répondre à trois besoins soit : une meilleure rigueur statistique, un indicateur de performance universel et finalement, une plus grande portée temporelle reflétant le caractère itératif des algorithmes étudiés. Voyons, en premier lieu, comment notre méthodologie répond à ces besoins.

5.1 Analyse statistique de la méthodologie

Comme mentionné précédemment, notre méthodologie ne permet pas toujours de déterminer qu'un algorithme est plus efficace qu'un autre, contrairement à une étude conventionnelle. Lorsque deux bandes sont superposées, nous ne pouvons affirmer qu'un algorithme est meilleur qu'un autre sous la condition du risque d'erreur contrôlé. Bien entendu, il est toujours possible de privilégier un algorithme par rapport à un autre même si leurs bandes se superposent. Toutefois, dans ces conditions le risque d'erreur devient supérieur au taux α choisi. Notre banc de test était constitué de 22 simulations. De ces 22 simulations, seulement

deux simulations (voir figures V.8 et figure V.23) ne sont pas caractérisées par au moins une superposition (d'au moins deux bandes) sur un intervalle temporel important (soit plus de 80% du domaine temporel). Cela signifie donc que dans la majorité des simulations de cette étude, il est impossible d'assumer, sous un risque d'erreur maximal de $\alpha = 0.05$, qu'il existe une différence statistiquement significative entre tous les échantillons. Il est évident qu'il est toujours plus intéressant de pouvoir tirer des conclusions claires permettant de classer en ordre de performance les algorithmes plutôt que d'avoir à gérer le « flou » entourant les superpositions d'intervalles de confiance. Toutefois, cela fait parti intégrante d'une étude statistique basée sur de petits échantillons de dimensions finis. Il y a toujours un risque d'erreur non nul, qui ne peut et ne doit être négligé.

Par contre, il existe différentes façons de minimiser le nombre de comparaisons dont les différences ne sont pas significatives. En augmentant le nombre d'observations d'un échantillon, nous raffinons les informations sur sa réelle distribution et implicitement, nous réduisons la largeur de sa bande de confiance. Nous augmentons ainsi nos chances d'éliminer la superposition de deux bandes sans même affecter le risque d'erreur. En second lieu, il est possible de tout simplement augmenter la valeur du seuil maximal d'erreur α. L'augmentation du seuil α aura pour effet d'amincir simultanément la bande de confiance de tous les échantillons de l'étude. Bien qu'il ne soit pas pertinent de travailler avec un seuil critique trop élevé, il peut être intéressant de vérifier pour quelle valeur minimale de α il est possible de différencier des échantillons. Il serait également pertinent de tenter d'augmenter la puissance statistique de la méthodologie dans des travaux futurs.

5.2 Utilisation du rang comme mesure de performance

L'utilisation du rang comme mesure de performance permettait principalement de pouvoir comparer toutes les instances sans discrimination sur l'atteinte de l'optimum et sans pondération subjective. Le seul cas où l'utilisation du rang n'est pas nécessaire pour garder l'objectivité des comparaisons, est celui où au maximum, une seule instance aurait atteint l'optimum global à l'intérieur de la fenêtre d'étude temporelle. Dans la présente étude, cela

ne s'est jamais produit. Aussi, cette situation est plutôt atypique, et il ne serait pas fondé de développer une méthodologie uniquement pour ce cas très particulier. De plus, dans ce cas précis, le seul impact négatif d'utiliser notre méthodologie est de ne pas exploiter la pleine puissance statistique disponible.

5.3 Analyse sur une étendue temporelle non ponctuelle

L'analyse sur un domaine temporel étendu permet à la fois de contenir l'équivalent d'un nombre élevé d'études ponctuelles (autant qu'il y a de comparaisons), d'associer un indice temporel à chacune de ces études et d'obtenir une « tendance » sur l'évolution des algorithmes. L'élément directeur de cette étude était fondé sur le principe qu'en tant que système stochastique, qu'un algorithme soit meilleur qu'un autre après un certain nombre d'itérations n'était pas une garantie pour un nombre d'itérations plus élevé. En observant les résultats obtenus, ce phénomène se reflèterait par un croisement des bandes de deux algorithmes dans le temps. Dans cette étude cela se produit 12 fois, soit dans 55% des cas. Ce ratio est immense si l'on considère qu'il est impossible de détecter ce phénomène lors d'une étude traditionnelle ponctuelle. Prenons par exemple le résultat de la figure V.13. Dans ce cas, pour les premiers deux tiers de la fenêtre temporelle, l'algorithme HS est supérieur à l'algorithme SHCLVND. Toutefois, à partir d'environ $7x10^5$ itérations, les rangs des algorithmes sont inversés très clairement et SHCLVND devient statistiquement supérieur à HS pour le reste de la fenêtre temporelle. Au moins deux études traditionnelles distinctes auraient été nécessaires pour faire ressortir ce phénomène. Ici, en plus de voir clairement son évolution, il est possible de connaître précisément les points critiques.

5.4 Analyse des trajectoires et des graphiques statistiques

À la lumière des résultats obtenus, il est intéressant de constater que les algorithmes possèdent tous des schémas d'évolution bien distincts. Par exemple, les algorithmes ES et PSO avaient des courbes d'évolution extrêmement rapides. En très peu d'itérations, ces deux algorithmes avaient habituellement soit atteint l'optimum global, ou étaient emprisonnés dans un optimum local. L'algorithme HS produisait des courbes d'évolution relativement

prononcées dans les premières itérations qui prenaient rapidement une pente très légère pour finalement suivre une asymptote imaginaire parallèle à l'axe des abscisses. Habituellement, si l'algorithme n'atteignait pas rapidement l'optimum global, il ne l'atteignait jamais (dans la fenêtre temporelle étudiée). L'algorithme DE avait une pente très forte pour les problèmes ayant une faible dimensionnalité et un petit nombre d'optimum locaux. Pour les problèmes plus complexes, il avait une courbe en deux temps avec une pente douce pour les premières itérations, et un point d'inflexion très prononcée suivi d'une pente très prononcée jusqu'à l'atteinte de l'optimum. Finalement, l'algorithme SHCLVND avait une courbe douce et régulière faisant en sorte que l'algorithme atteignait dans la majorité des cas l'optimum global, mais cela, au prix d'un nombre élevé d'itérations.

L'algorithme le plus efficace de cette étude est sans contredit l'évolution différentielle. L'algorithme fut significativement meilleur que les autres sur presque la totalité du domaine temporel dans 11 des 22 simulations de notre banc de test. De plus, l'algorithme était parmi les deux meilleurs algorithmes dans 5 autres simulations. L'algorithme est également celui ayant le meilleur taux d'atteinte de l'optimum.

L'algorithme ES fut particulièrement efficace pour les problèmes à deux dimensions ayant un petit nombre d'optimum locaux. Ainsi, il fut le meilleur algorithme sur presque l'ensemble du domaine temporel pour les problèmes *Rcos de Branin*, *Six humps camel* et *Goldtein-Price*. Il fut toutefois surprenamment inefficace pour le problème *Shekel's foxholes*.

L'algorithme PSO, arriva à des résultats similaires à l'algorithmes ES. Il était toutefois légèrement moins performant pour les problèmes dans lesquelles ES excellait mais plus performant pour les problèmes plus complexes. Il fut le meilleur algorithme pour le problème *Rosenbrock's Saddle*, et fut l'un des deux meilleurs algorithmes pour les problèmes Hyper-ellipsoïde à 30 dimensions et *Griewangk* à 2 et 30 dimensions.

L'algorithme HS fut l'algorithme le moins efficace de cette étude. Mis à part pour le problème *Shekel's foxholes*, pour lequel il fut le meilleur algorithme (mais non

significativement différentiable de l'algorithme DE), ses performances furent moyennes et même médiocres pour les problèmes complexes. L'algorithme n'a atteint que rarement l'optimum global. Il est toutefois intéressant de préciser que l'algorithme fut développé pour des applications bien précises, soit pour optimiser le déploiement d'un réseau d'aqueduc de grandes villes ou pour résoudre des problèmes structuraux. Il est donc possible de croire que l'algorithme soit hautement spécialisé et qu'aucun des problèmes de notre banc de test ne corresponde au type de problème pour lequel cet algorithme fut initialement développé.

Finalement, les résultats de l'algorithme SHCLVND ne sont pas très bons. Bien que l'algorithme eue un taux d'atteinte de l'optimum global relativement élevé, l'algorithme ne convergeait pas assez rapidement pour être compétitif, si bien qu'il fut le pire algorithme de l'étude pour plusieurs problèmes.

CONCLUSION

Les métaheuristiques pour l'optimisation sont des algorithmes pouvant être très efficaces. Toutefois, la spécificité de ces algorithmes, le théorème NFL et le fait qu'ils soient itératifs et non déterministes, rend complexe l'évaluation de leurs performances. Cela dit, le choix d'un algorithme et de son paramétrage est le principal défi d'un utilisateur confronté à un problème d'optimisation. La caractérisation des performances de ces algorithmes représente donc un enjeu important dans l'amélioration des performances de ces méthodes, de la compréhension de leur fonctionnement et de leur facilité d'utilisation.

Ce travail avait pour objectif de développer une méthodologie de comparaison des performances des métaheuristiques à la fois représentative, statistiquement fiable, portable et facile d'utilisation. L'atout principal de la méthode proposée est l'utilisation d'une statistique de rang. En effet, l'utilisation du rang a permis d'éviter d'utiliser des coefficients arbitraires lors de l'évaluation de la qualité des processus. Il est donc possible de considérer à la fois les individus ayant atteint l'optimum global et ceux ne l'ayant pas atteint sous une même mesure de qualité équitable. Toutefois, cet atout engendre également une perte de puissance statistique puisque l'utilisation des rangs ne permet pas d'établir d'amplitude entre la qualité des individus. Cette faiblesse statistique peut être contournée en augmentant le nombre d'échantillons. Dans cette étude, il a été déterminé que 50 échantillons par algorithme permettaient d'atteindre une puissance statistique satisfaisante. Une autre innovation intéressante de ce travail était de considérer le nombre d'itérations comme un paramètre et de superposer de multiples variantes de ce paramètre sur une seule image. La lecture des résultats est ainsi très simplifiée. Toutefois, la méthode de comparaison permet d'obtenir une note globale d'un algorithme englobant plusieurs facteurs. La méthode de comparaison est ainsi un outil supplémentaire fort pertinent mais qui n'élimine pas l'utilisation de mesures telles que le taux d'atteinte de l'optimum, et le nombre d'itérations moyen pour l'atteindre.

L'utilisation du nombre d'évaluations comme indicateur temporel introduit également un léger biais favorisant les algorithmes dont le processus générateur est plus complexe. Cette

façon de faire possède par contre l'avantage de dissocier les ressources matérielles, le langage de programmation utilisé et le niveau d'optimisation temporelle du code de l'analyse. Un utilisateur pourrait aussi aisément mesurer le temps d'exécution de chaque algorithme sous son environnement matériel et selon son code source pour ajouter des coefficients permettant d'ajuster l'analyse selon ses propres besoins.

Quoique secondaire, ce travail a également permis de bien situer l'importance du banc de test et de souligner le lien qu'il existe entre les résultats obtenus et les problèmes du banc de test utilisé. En somme, ce travail aura permis de développer une méthodologie de comparaison universelle et statistiquement rigoureuse permettant d'étudier les performances des métaheuristiques appliquées à l'optimisation de problèmes continus monocritères.

RECOMMANDATIONS

Bien que la partie statistique de la méthodologie puisse difficilement être modifiée tout en conservant son caractère objectif et non paramétrique, il serait possible d'apporter quelques modifications à la procédure pour la rendre plus efficace.

Dans un premier temps, il serait possible d'éliminer pratiquement toutes les égalités de rang entre les instances. Théoriquement, il est possible, mais très peu probable, que deux trajectoires soient identiques. Bien entendu, plus le nombre d'itérations des trajectoires est élevé, plus cette possibilité est faible. Aussi, il est évident que deux trajectoires strictement identiques ne pourront pas être différentiées pour aucun paramétrage temporel. Toutefois, il est toujours possible de donner un rang unique à deux trajectoires uniques. Considérons les deux cas possibles de notre étude, soit deux trajectoires n'ayant pas atteint l'optimum global et ayant exactement la même valeur de coût pour le paramétrage temporel étudié, et deux trajectoires ayant atteint l'optimum global au même moment (même nombre d'itérations).

Dans le premier cas, il est possible de faire une distinction entre les deux trajectoires en favorisant celle dont la valeur de coût était la meilleure à l'itération précédente. En cas d'égalité, il suffit de réitérer le processus jusqu'à ce que les trajectoires n'aient plus la même valeur de coût.

Pour le second cas, il est possible de comparer les valeurs de coût des trajectoires pour l'itération précédant celle de l'atteinte de l'optimum. En cas d'égalité, il suffit encore une fois de refaire le procédé pour l'itération précédente jusqu'à ce qu'il ait une différence.

La méthode, étant statistiquement rigoureuse sous un risque d'erreur contrôlé, ne permet pas toujours de conclure qu'un algorithme est meilleur qu'un autre. Comme mentionnée précédemment à la section 5.1, il est toutefois possible de réduire le niveau d'incertitude en augmentant le nombre d'observations d'un échantillon. L'augmentation du nombre d'observations aura pour effet de réduire l'intervalle de confiance d'un échantillon et

rehaussera ainsi la probabilité de pouvoir faire une distinction significative entre deux échantillons. Suivant cette directive, il serait possible de faire un échantillonnage dynamique des algorithmes en incorporant une partie d'analyse des résultats et de gestion intelligente de l'échantillonnage de manière à maximiser la portée des résultats pour un temps de calcul donné. La largeur des bandes de confiances ne serait donc plus statiques et uniformes mais plutôt variable au fils des itérations. Dans la même optique, il serait possible d'envelopper la méthodologie d'un système autonome complexe permettant d'ajuster un algorithme pour le rendre très performant pour un type de problème donné. En utilisant les informations issues de la méthodologie d'analyse, le système ajusterait automatiquement les paramètres de l'algorithme, un peu comme un système d'apprentissage pour des systèmes d'intelligence artificielle

La méthodologie proposée dans cette étude fut développée et validée pour des problèmes d'optimisations continus monocritères. Rien n'empêche malgré tout de l'utiliser pour comparer les performances de métaheuristiques appliquées à des problèmes combinatoires ou discrets. En fait, du point de vue de la méthodologie, il n'existe aucune distinction entre les deux types de problèmes.

Il est toutefois plus complexe d'étendre la méthodologie à des problèmes multicritères tout en préservant l'objectivité de l'étude et la généralité des résultats. Il serait possible de conjuguer tous les critères sous un seul critère par une sommation pondérée. Cela permettrait, du point de vue de la méthodologie, de convertir un problème multicritère en un problème monocritère de manière synthétique. Cette façon de faire possède toutefois le net désavantage de réduire la validité des résultats à une seule fonction de synthétisation des critères. Les résultats ne seraient donc plus génériques mais intimement reliés à la fonction de sommation utilisée pour synthétiser les critères. À l'opposé, il serait également possible de séparer tous les critères et d'effectuer une étude comparative distincte pour chacun d'eux. En procédant de la sorte il serait intéressant de développer une façon de joindre les résultats de chaque critère d'une manière statistiquement fiable et indépendante de pondérations quelconques ou du moins, qui permettrait de connaître l'impact des pondérations utilisées.

ANNEXE I

VALEURS CRITIQUES DU TEST DE L'ÉTENDUE DE STUDENT $Q_{k,\infty}^{(\alpha)}$

Le test de l'étendue de Student permet de

Tableau I.1 Valeurs critiques du test de l'étendue de Student pour $\nu = \infty$

k \ α	**0.10**	**0.05**	**0.01**
2	2.326	2.772	3.643
3	2.902	3.314	4.120
4	3.240	3.633	4.403
5	3.478	3.858	4.603
6	3.661	4.030	4.757
7	3.808	4.170	4.882
8	3.931	4.286	4.987
9	4.037	4.387	5.078
10	4.129	4.474	5.157
11	4.211	4.552	5.227
12	4.285	4.622	5.290
13	4.351	4.685	5.348
14	4.412	4.743	5.400
15	4.468	4.796	5.448
16	4.519	4.845	5.493
17	4.568	4.891	5.535
18	4.612	4.934	5.574
19	4.654	4.974	5.611
20	4.694	5.012	5.645

ANNEXE II

DISTRIBUTION DU CHI-CARRÉ

La distribution de probabilité du χ^2 est est un cas spécial de la fonction Γ et suit l'équation :

$$y = f(x|\nu) = \frac{x^{(\nu-2)/2} e^{-x/2}}{2^{\frac{\nu}{2}} \Gamma(\nu/2)}$$

Où ν est le nombre de degrés de liberté et $\Gamma(\cdot)$, la fonction Gamma.

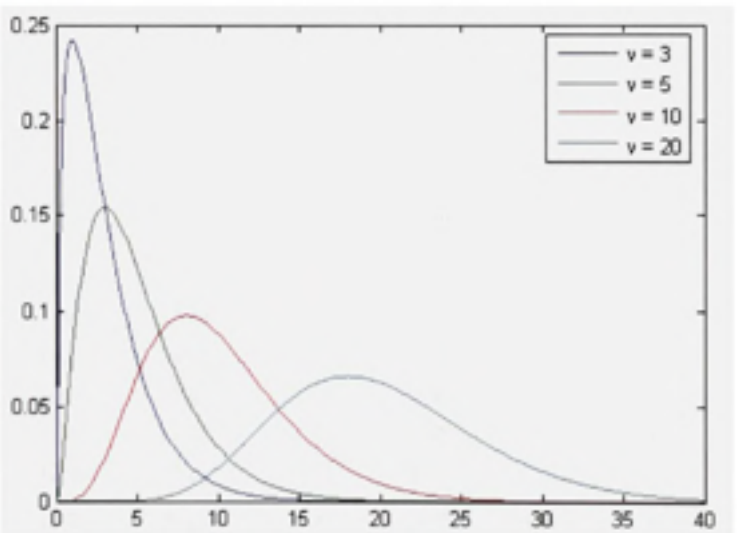

Figure II.1 ***Distributions du Chi-carré pour quatre degrés de liberté***

La distribution cumulative de la fonction du χ^2 suit l'équation suivante :

$$p = F(x|\nu) = \int_0^x \frac{t^{(\nu-2)/2} e^{-t/2}}{2^{\frac{\nu}{2}} \Gamma(\nu/2)} dt$$

Où $\Gamma(\cdot)$ est la fonction Gamma et p la probabilité qu'une seule observation d'une distribution du χ^2 à ν degrés de liberté se situe dans l'interval $[0\ x]$.

Tableau II.2 Probabilité d'extrémité droite de distribution du Chi-carré

ν \ p	**0.95**	**0.90**	**0.50**	**0.25**	**0.10**	**0.05**	**0.01**	**0.005**	**0.001**
1	0.004	0.016	0.45	1.32	2.71	3.84	6.63	7.88	10.83
2	0.10	0.21	1.39	2.77	4.61	5.99	9.21	10.60	13.82
3	0.35	0.58	2.37	4.11	6.25	7.81	11.34	12.84	16.27
4	0.71	1.06	3.36	5.39	7.78	9.49	13.28	14.86	18.47
5	1.15	1.61	4.35	6.63	9.24	11.07	15.09	16.75	20.52
6	1.64	2.20	5.35	7.84	10.64	12.59	16.81	18.55	22.46
7	2.17	2.83	6.35	9.04	12.02	14.07	18.48	20.28	24.32
8	2.73	3.49	7.34	10.22	12.36	15.51	20.09	21.96	26.12
9	3.33	4.17	8.34	11.39	14.68	16.92	21.67	23.59	27.88
10	3.94	4.87	9.34	12.55	15.99	18.31	23.21	25.19	29.59
11	4.57	5.58	10.34	13.70	17.28	19.68	24.72	26.76	31.26
12	5.23	6.30	11.34	14.85	18.55	21.03	26.22	28.30	32.91
13	5.89	7.04	12.34	15.98	19.81	22.36	27.69	29.82	34.53
14	6.57	7.79	13.34	17.12	21.06	23.68	29.14	31.32	36.12
15	7.26	8.55	14.34	18.25	22.31	25.00	30.58	32.80	37.70
16	7.96	9.31	15.34	19.37	23.54	26.30	32.00	34.27	39.25
17	8.67	10.09	16.34	20.49	24.77	27.59	33.41	35.72	40.79
18	9.39	10.86	17.34	21.60	25.99	28.87	34.81	37.16	42.31
19	10.12	11.65	18.34	22.72	27.20	30.14	36.19	38.58	43.82
20	10.85	12.44	19.34	23.83	28.41	31.14	37.57	40.00	45.32
21	11.59	13.24	20.34	24.93	29.62	32.67	38.93	41.40	46.80
22	12.34	14.04	21.34	26.04	30.81	33.92	40.29	42.80	48.27
23	13.09	14.85	22.34	27.14	32.01	35.17	41.64	44.18	49.73
24	13.85	15.66	23.34	28.24	33.20	36.42	42.98	45.56	51.18
25	14.61	16.47	24.34	29.34	34.38	37.65	44.31	46.93	52.62
26	15.38	17.29	25.34	30.43	35.56	38.89	45.64	48.29	54.05
27	16.15	18.11	26.34	31.53	36.74	40.11	46.96	49.64	55.48
28	16.93	18.94	27.34	32.62	37.92	41.34	48.28	50.99	56.89
29	17.71	19.77	28.34	33.71	39.09	42.56	49.59	52.34	58.30
30	18.49	20.60	29.34	34.80	40.26	43.77	50.89	53.67	59.70

ANNEXE III

AJUSTEMENT DES FONCTIONS DE TEST

Tableau III.3 Ajustement des fonctions de test

Fonction	d	ζ	$\overline{tr}$
De Jong #1	2	-	[−2.2773 −4.3679]
	30	-	[3.7201 −2.4988 −0.4784 0.6443 2.7476 -1.2651 -2.4042 0.2388 2.9493 -2.9322 -3.4185 -2.2956 2.5742 2.0742 -0.2795 -2.7318 3.1636 4.2886 -2.0410 3.9144 -0.3543 1.9488 -1.8275 1.0100 3.3733 -1.0371 -2.3580 -1.5401 3.6408 -1.8890]
Hyper-ellipsoïde	2	-	[-0.8183 −1.0681]
	30	-	[0.2398 2.0288 -0.6341 3.8139 3.1730 2.2845 1.9716 3.6450 -1.0522 -1.8039 -4.3979 -4.2824 3.4777 2.7803 -2.9675 -0.6967 4.3169 2.3742 -3.8728 3.5550 -0.9140 1.6369 3.0677 -1.3341 -0.1348 -1.4828 1.1118 3.9671 -4.5687 3.6031]
De Jong #2	2	-	-
Rcos de Branin	2	≈ −0.39789	-
Six humps camel	2	≈ 1.03163	-
Goldstein-Price	2	−3	-
De Jong #5	2	≈ −0.99800	-
Rastrigin généralisée	2	-	[432.2469 260.4254]
	30	-	[80.173 96.865 354.47 −288.39 309.62 161.23 −259.78 414.04 −294.75 20.699 108.65 −145.11 −104.26 393.13 378.12 −192.87 −301.88 −488.71 210.43 −535.35 384.36 −298.31 −73.888 225.22 406.99 156.80 −30.084 330.81 −158.53 −323.61]
Ackley	2	-	[2.4044 −10.0080]
	30	-	[−2.8549 21.974 −19.972 −9.6473 24.623 −17.640 −20.496 −8.8270 −19.119 −13.884 3.2986 5.7240 3.1853 −2.3892 15.852 8.6927 −18.936 13.562 20.252 4.5670 2.6607 −18.562 −4.9008 −13.020 −20.850 8.8909 −4.0242 −0.2696 −10.190 −10.950]

Tableau III.3 Ajustement des fonctions de test (suite)

Fonction	d	ζ	$\overline{tr}$
Shubert pénalisée	2		-
Corana's parabola	2	-	[−288.6620 −285.3618]
	30	-	[−255.06 816.72 −314.87 −869.01 572.62 177.09 −854.65 186.24 −371.57 −407.33 652.73 118.84 −68.23 618.38 129.58 −749.98 429.18 −177.05 522.63 467.06 −231.41 119.14 351.51 104.40 154.33 194.04 −513.33 3.8375 −324.83 −543.39]
Griewangk	2	-	[−223.80 −125.42]
	30	-	[−149.48 465.43 −261.80 530.28 −188.01 −253.74 −34.563 87.438 536.78 358.30 29.774 103.20 −427.84 128.71 2.4383 352.90 −516.08 −75.307 −361.88 −167.96 −473.25 −170.24 −301.31 1.7804 −260.82 −96.510 −108.05 −325.22 −483.43 415.76]
Schwefel	2	≈ 837.966	-
	30	≈ 12569.487	-
Michalewicz	2	≈ 1.80130	-
	30	≈ 29.6309	-

ANNEXE IV

CARACTÉRISATION DES FONCTIONS DE TEST

Tableau IV.4 Caractéristiques des tests

Fonction	d	#mins	Épistatie	C	Régularité	Isolement	Continue
De Jong #1	2 30	1	nulle	100%	-	-	oui
Hyper-ellipsoïde	2 30	1	nulle	100%	-	-	oui
De Jong #2	2	1	forte	100%	-	-	oui
Rcos de Branin	2	3[1]	forte	100%	-	-	oui
Six humps camel	2	6[2]	faible	≈1%	faible	faible	oui
Goldstein-Price	2	4	moyenne	≈0.41%	aucune	moyen	oui
De Jong #5	2	25	faible	≈10%	forte	faible	oui
Rastrigin généralisée	2 30	1201[4]	nulle	≈7×10^{-5}% ≈1.6×10^{-29}%	forte	nul	oui
Ackley	2 30	61[4]	nulle	≈1.4×10^{-2}% ≈2.3×10^{-9}%	forte	nul	oui
Shubert pénalisée	2	760	nulle	≈0.5%	forte	faible	oui
Corana's parabola	2 30	10001[4]	nulle	2.25×10^{-6}% ≈1.918×10^{-113}%	forte	nul	non
Griewangk	2 30	- -	nulle	- -	forte	nul	oui
Schwefel modifiée	2 30	7[4]	nulle	≈3.92% ≈9.3×10^{-10}%	forte	moyen	Oui[3]
Michalewicz	2 30	d!	nulle	≈7.1% -	Très faible	moyen	oui

[1] Les trois sont globaux

[2] Deux sont globaux

[3] Une discontinuité aux bornes

ANNEXE V

RÉSULTATS

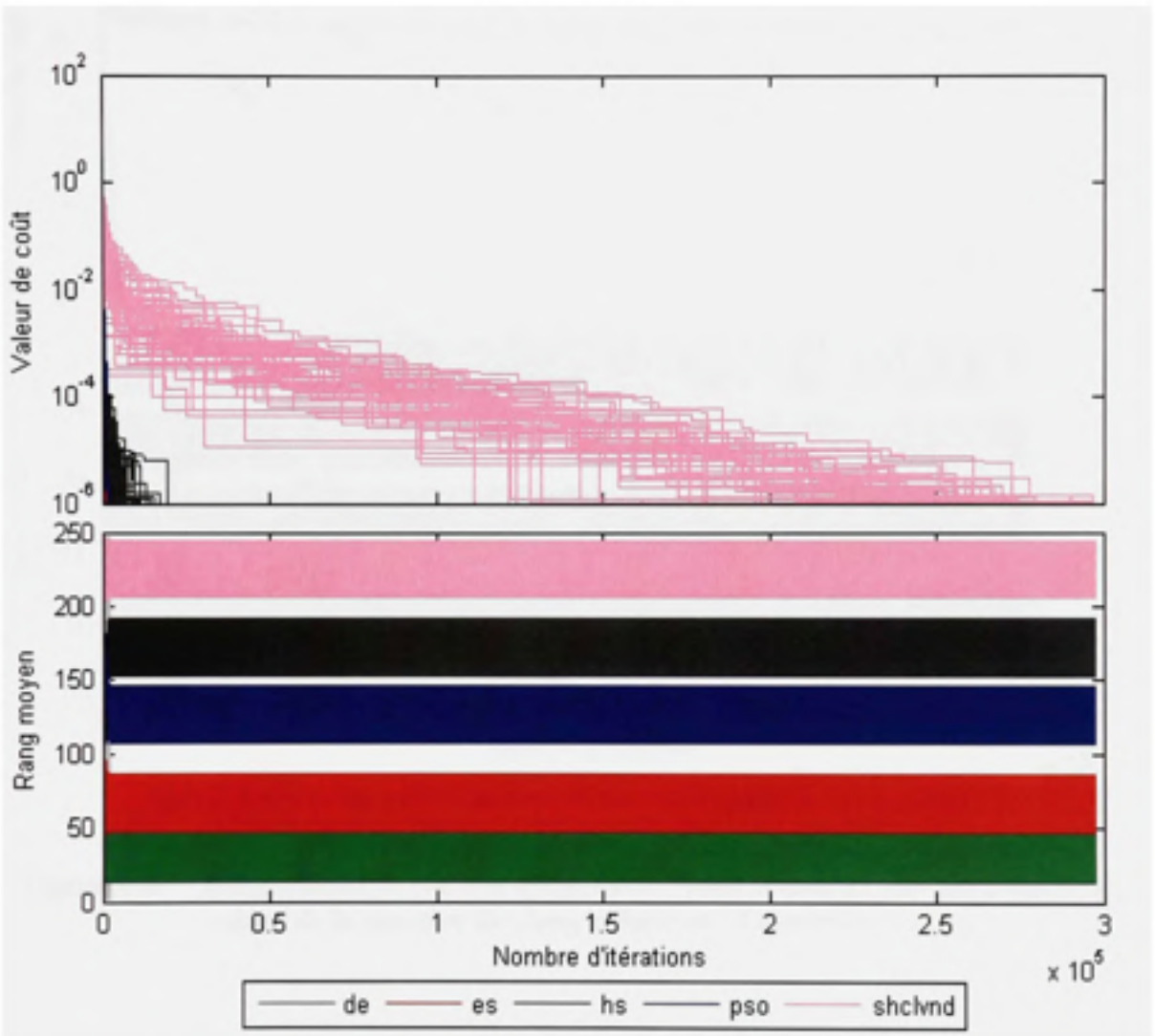

Figure V.2 ***Échantillons de courbes d'évolution (haut) et analyse des performances (bas) de la fonction De Jong sphère en 2 dimensions.***

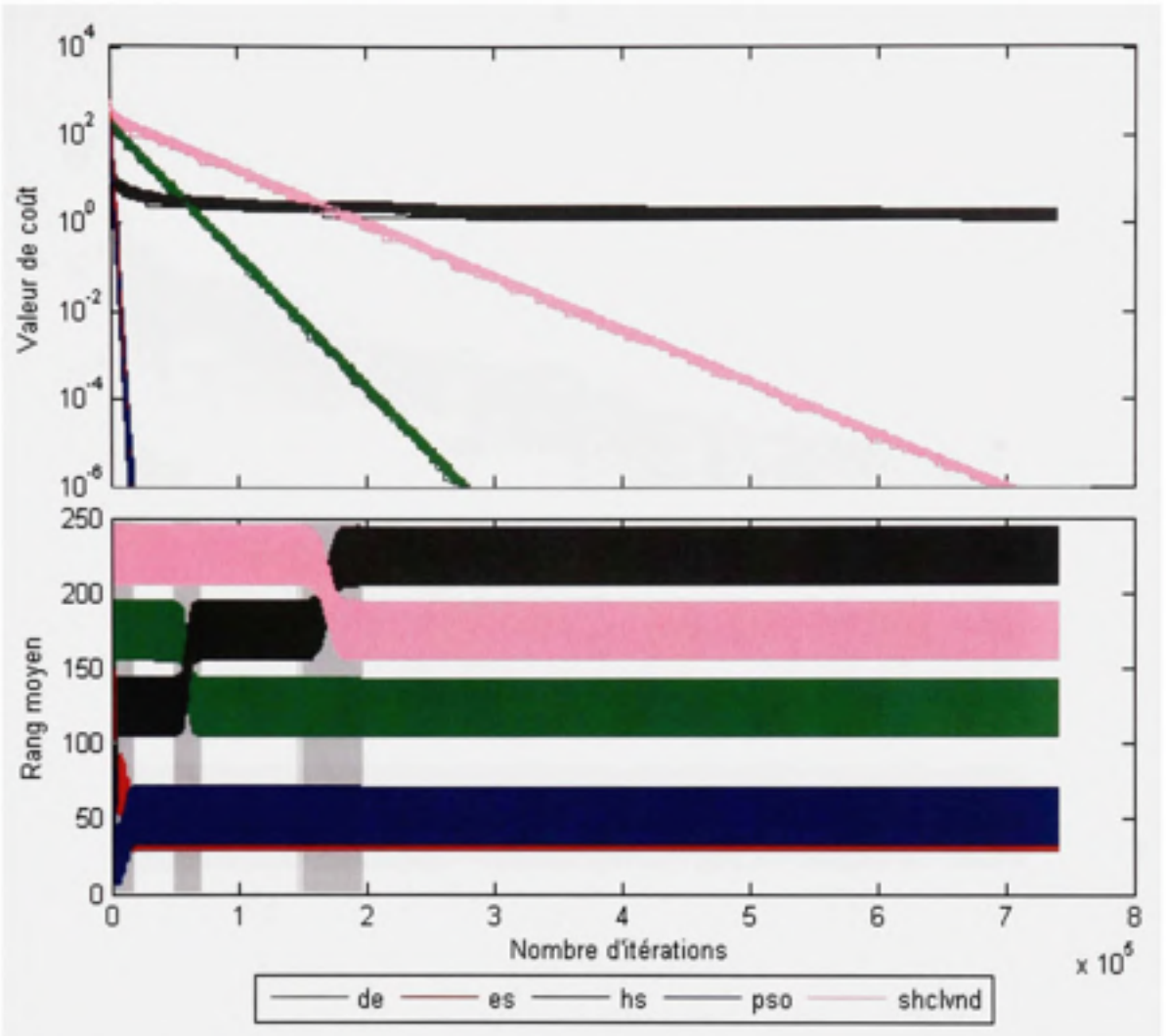

Figure V.3 ***Échantillons de courbes d'évolution (haut) et analyse des performances (bas) de la fonction De Jong sphère en 30 dimensions.***

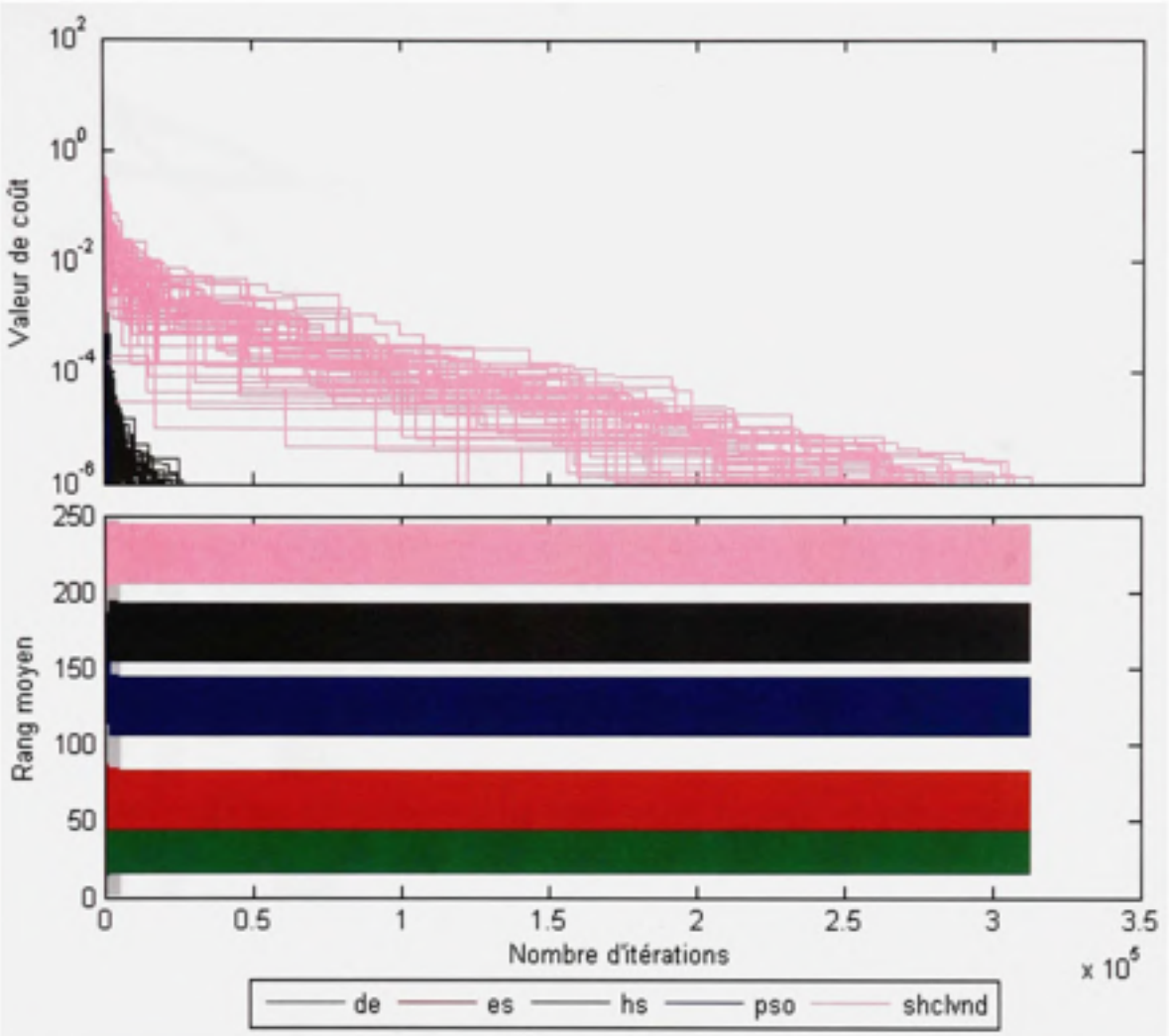

Figure V.4 ***Échantillons de courbes d'évolution (haut) et analyse des performances (bas) de la fonction Hyper-ellipsoïde en 2 dimensions.***

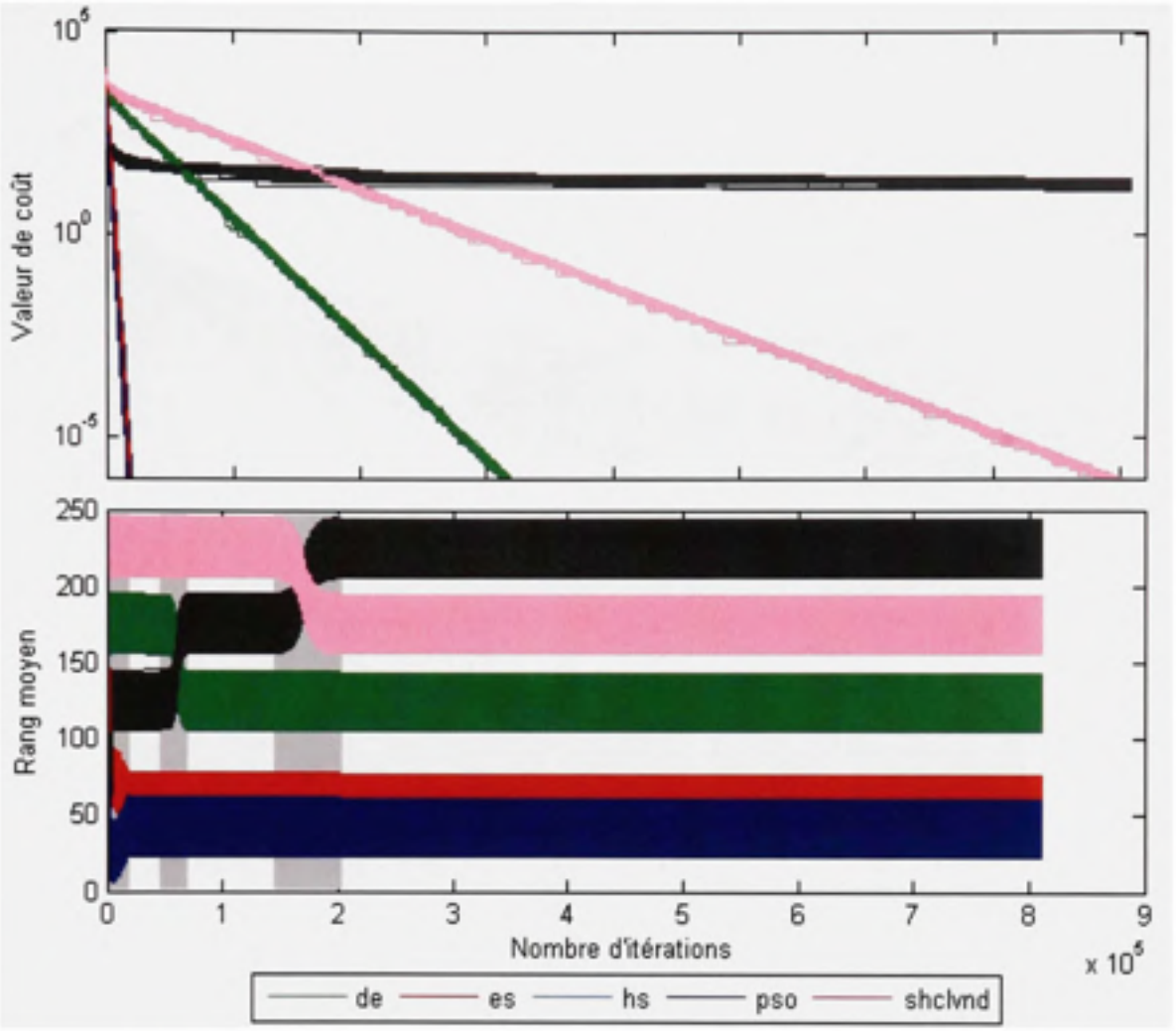

Figure V.5 ***Échantillons de courbes d'évolution (haut) et analyse des performances (bas) de la fonction Hyper-ellipsoïde en 30 dimensions.***

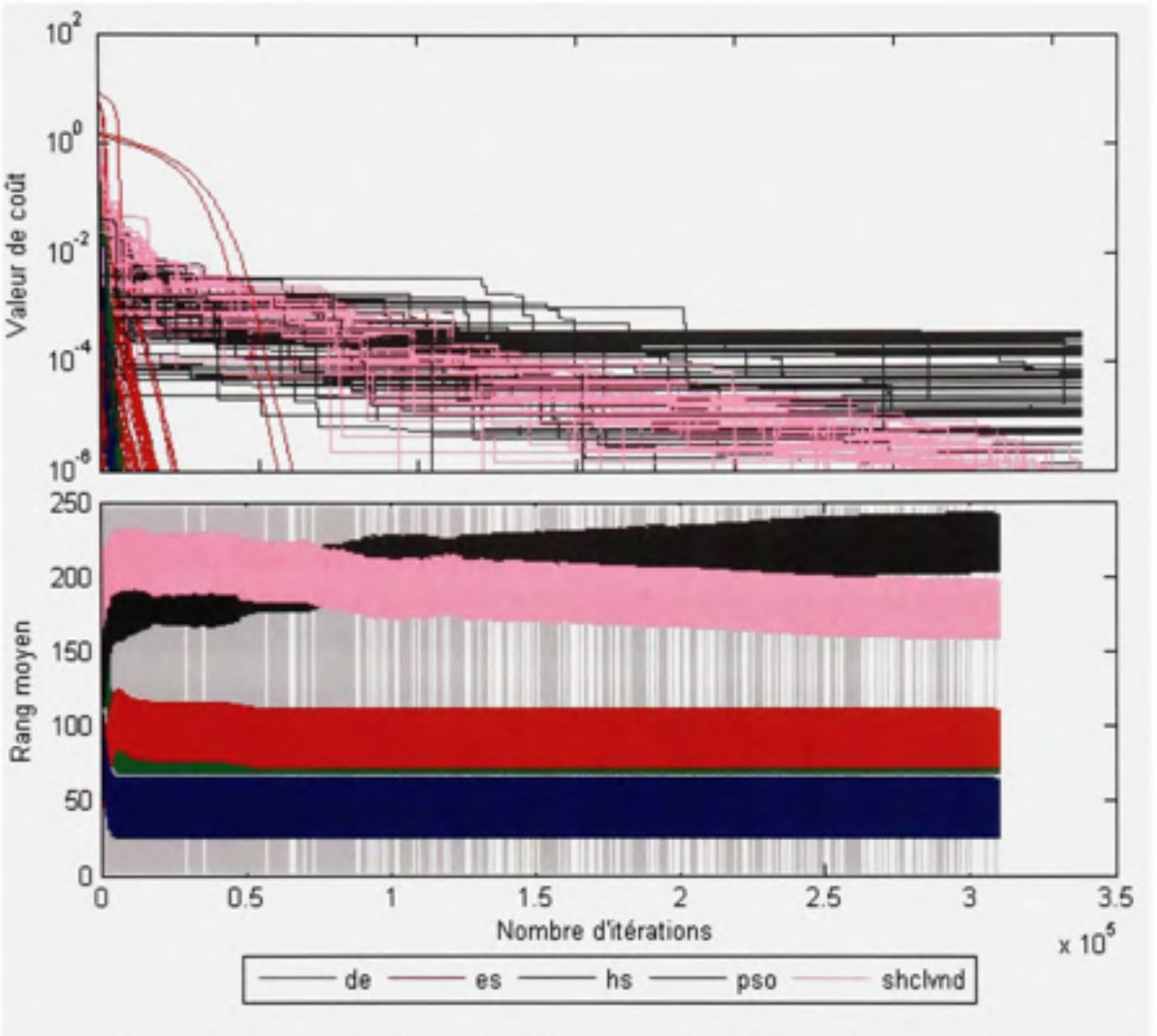

Figure V.6 ***Échantillons de courbes d'évolution (haut) et analyse des performances (bas) de la fonction Rosenbrock's saddle en 2 dimensions.***

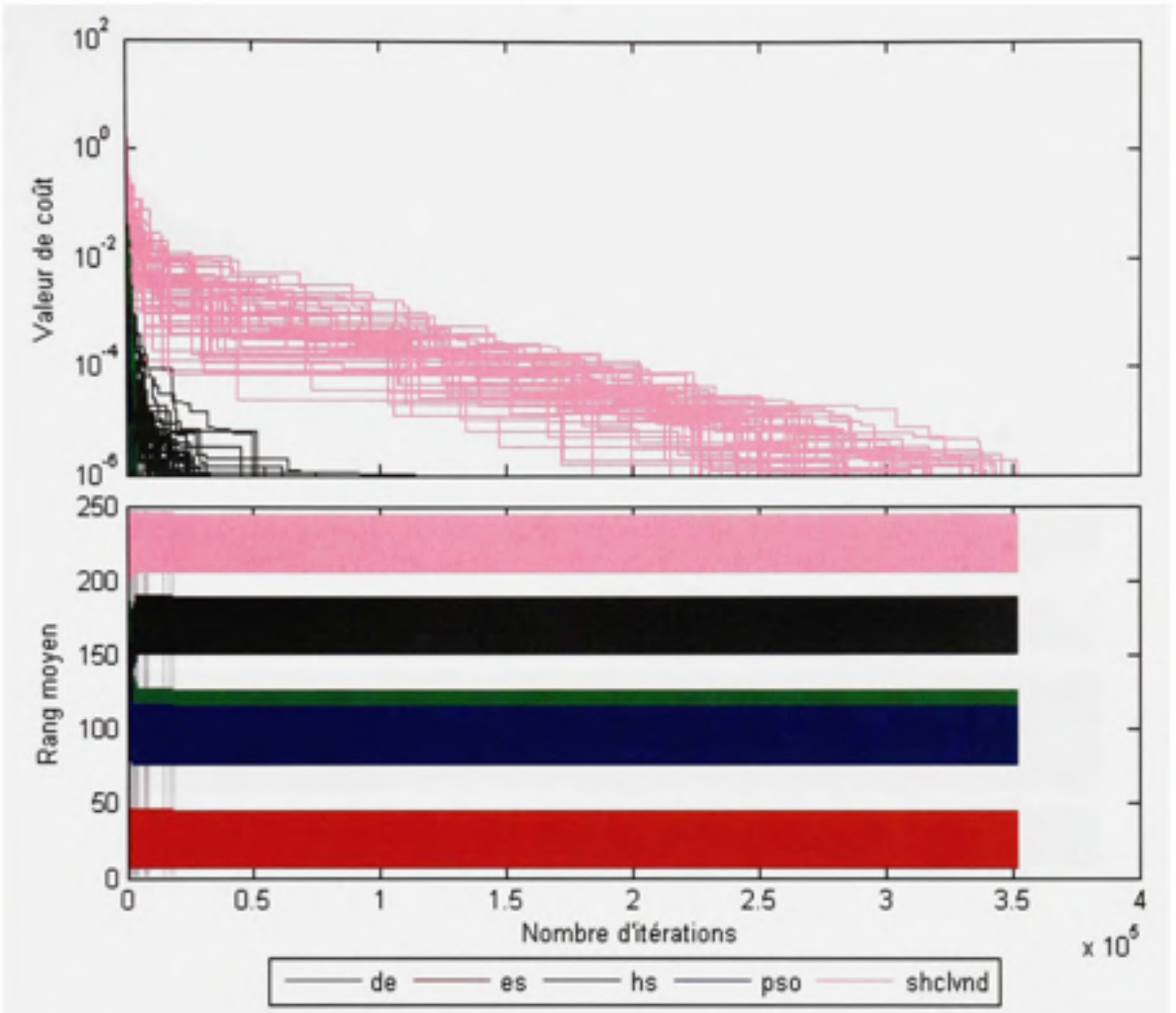

Figure V.7 ***Échantillons de courbes d'évolution (haut) et analyse des performances (bas) de la fonction Rcos de Branin en 2 dimensions.***

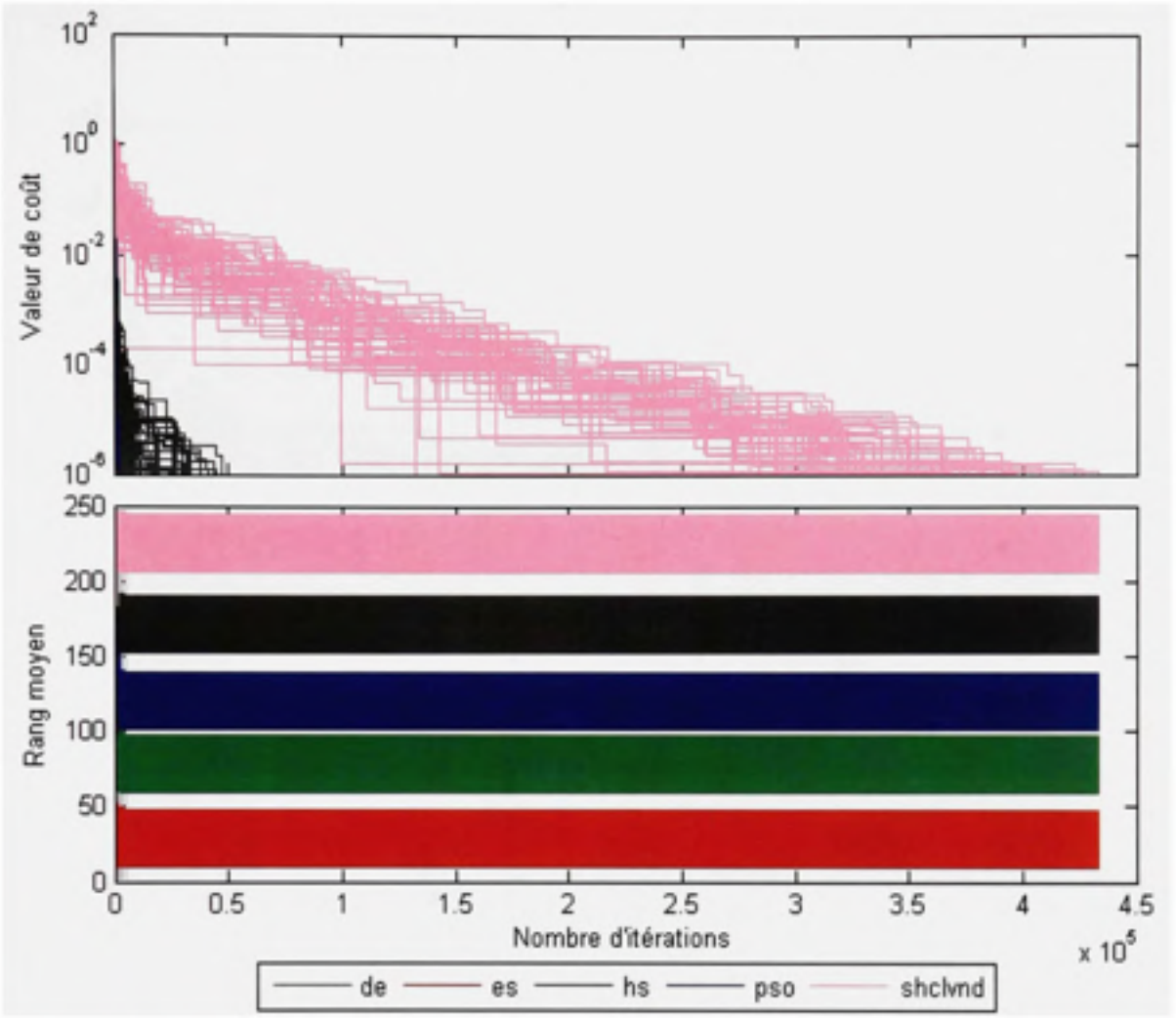

Figure V.8 ***Échantillons de courbes d'évolution (haut) et analyse des performances (bas) de la fonction Six humps camel en 2 dimensions.***

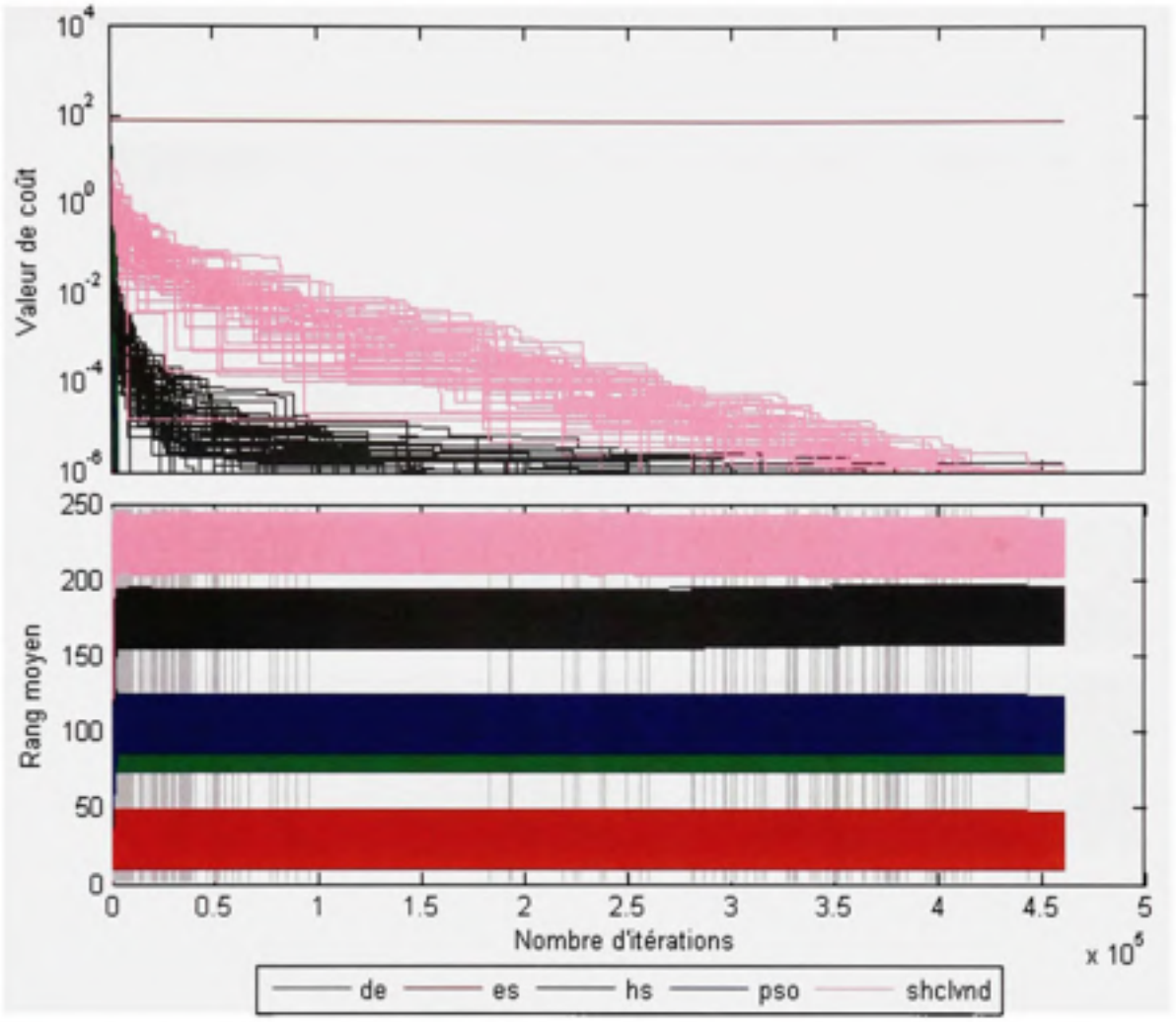

Figure V.9 ***Échantillons de courbes d'évolution (haut) et analyse des performances (bas) de la fonction Goldstein-Price en 2 dimensions.***

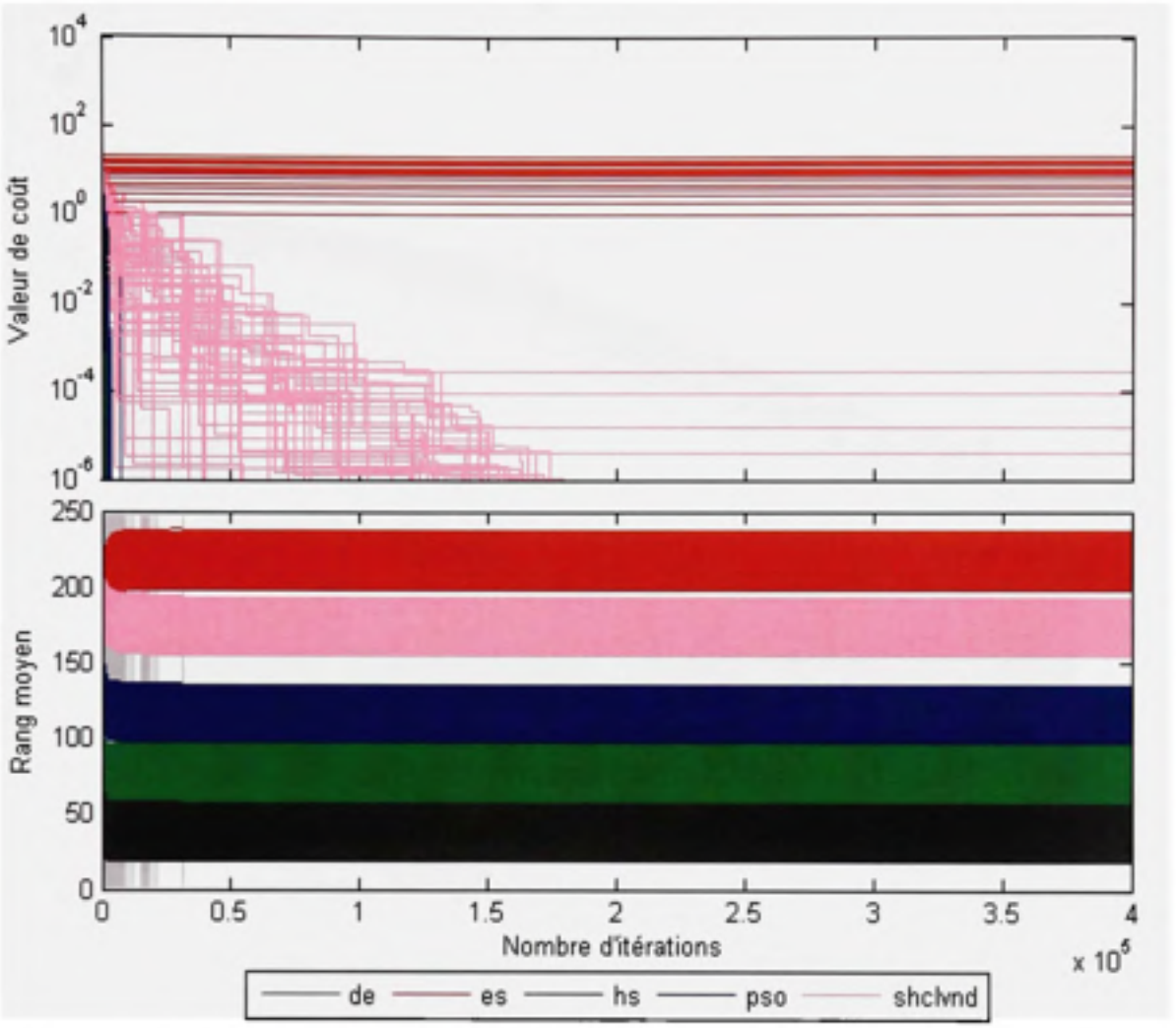

Figure V.10 ***Échantillons de courbes d'évolution (haut) et analyse des performances (bas) de la fonction Shekel's foxholes en 2 dimensions.***

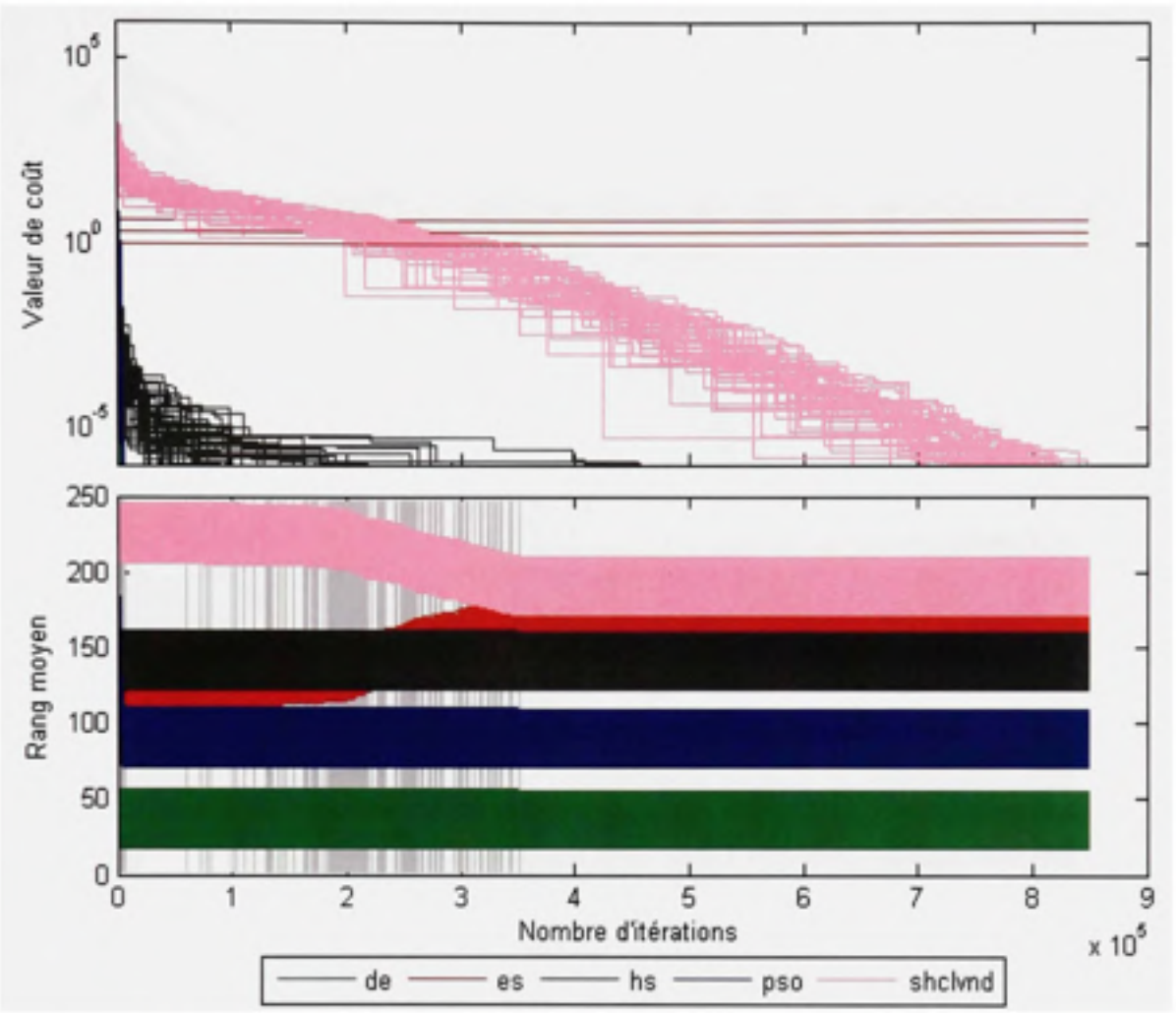

Figure V.11 ***Échantillons de courbes d'évolution (haut) et analyse des performances (bas) de la fonction Rastrigin généralisée en 2 dimensions.***

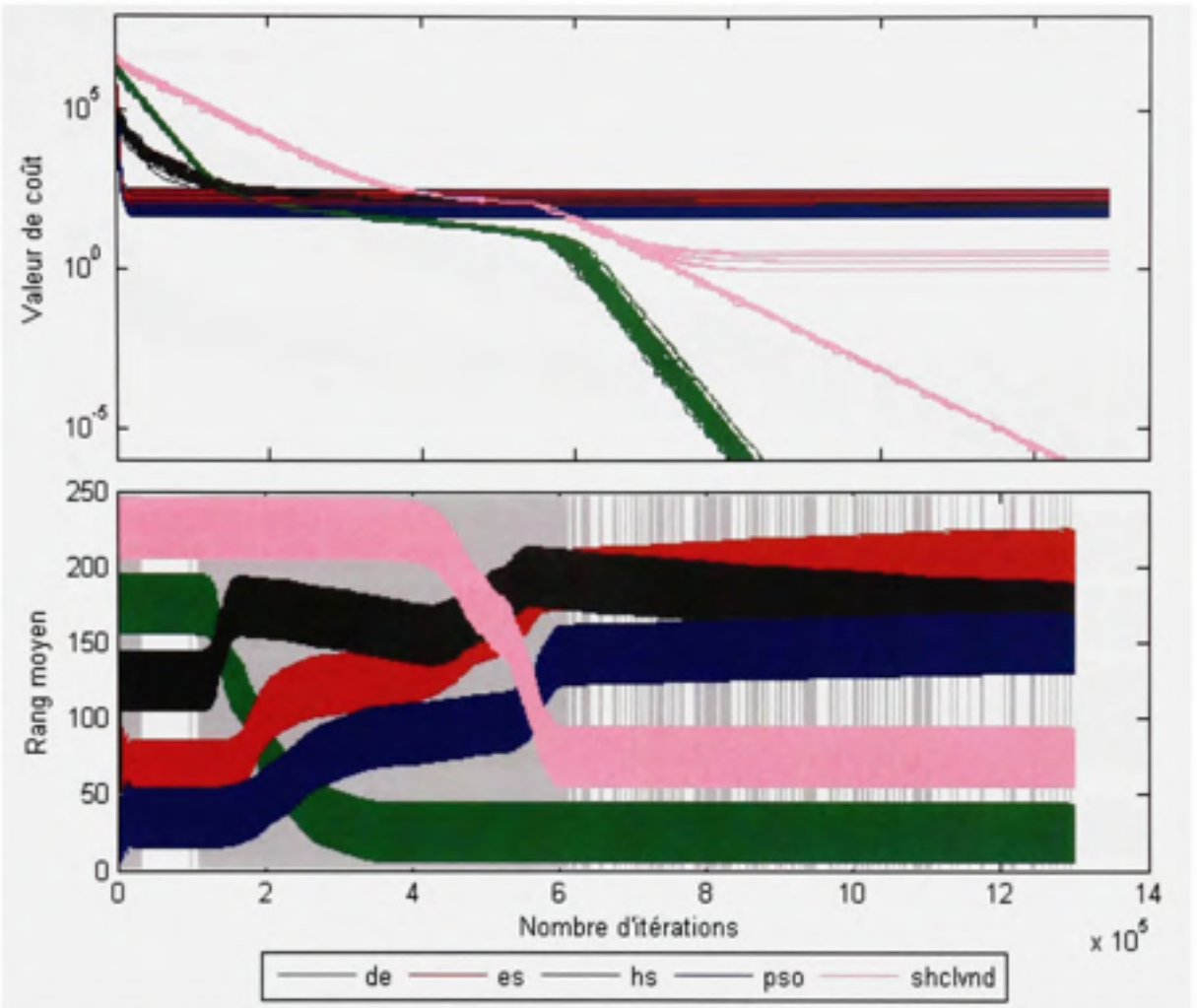

Figure V.12 ***Échantillons de courbes d'évolution (haut) et analyse des performances (bas) de la fonction Rastrigin généralisée en 30 dimensions.***

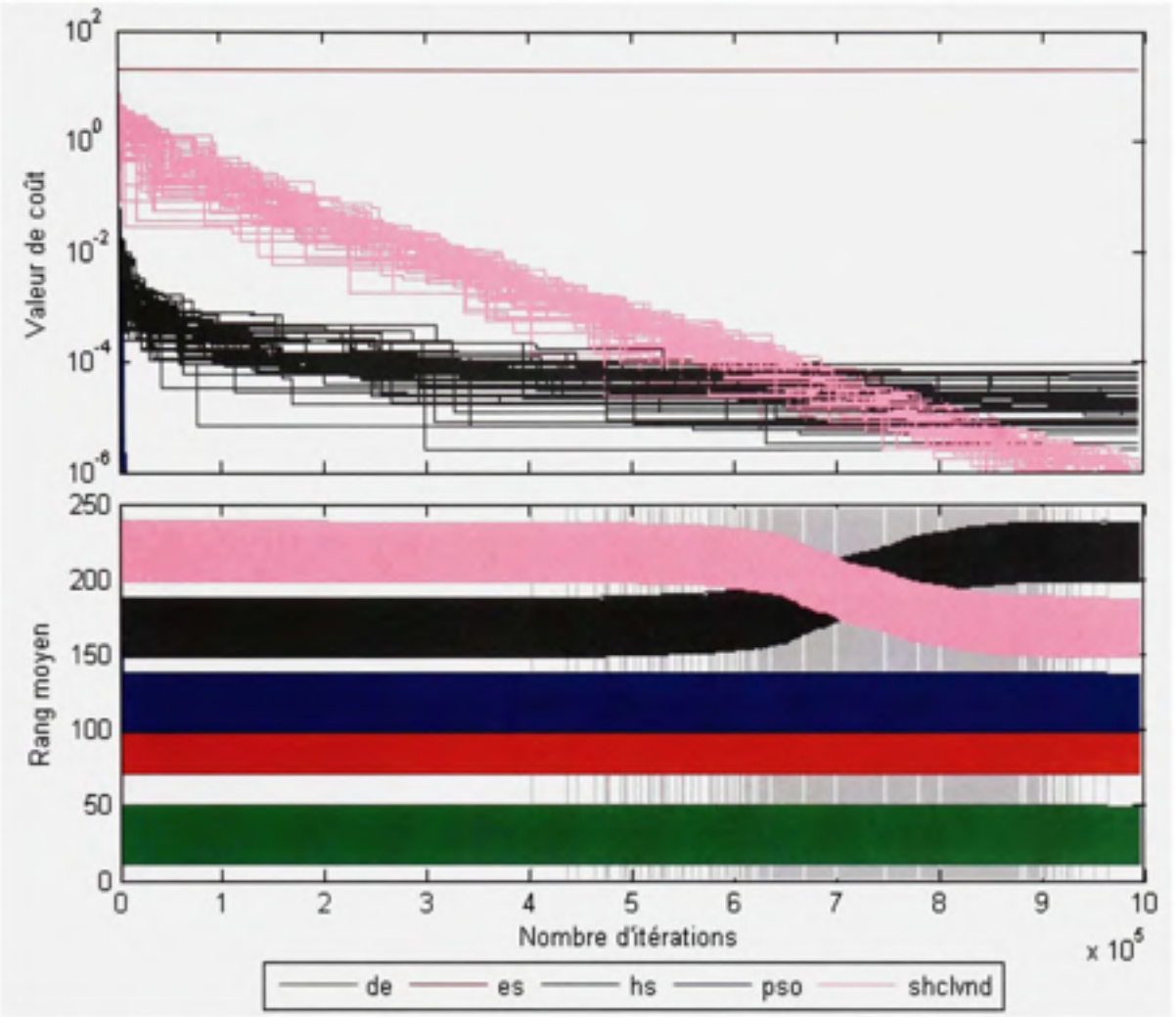

Figure V.13 ***Échantillons de courbes d'évolution (haut) et analyse des performances (bas) de la fonction Ackley en 2 dimensions.***

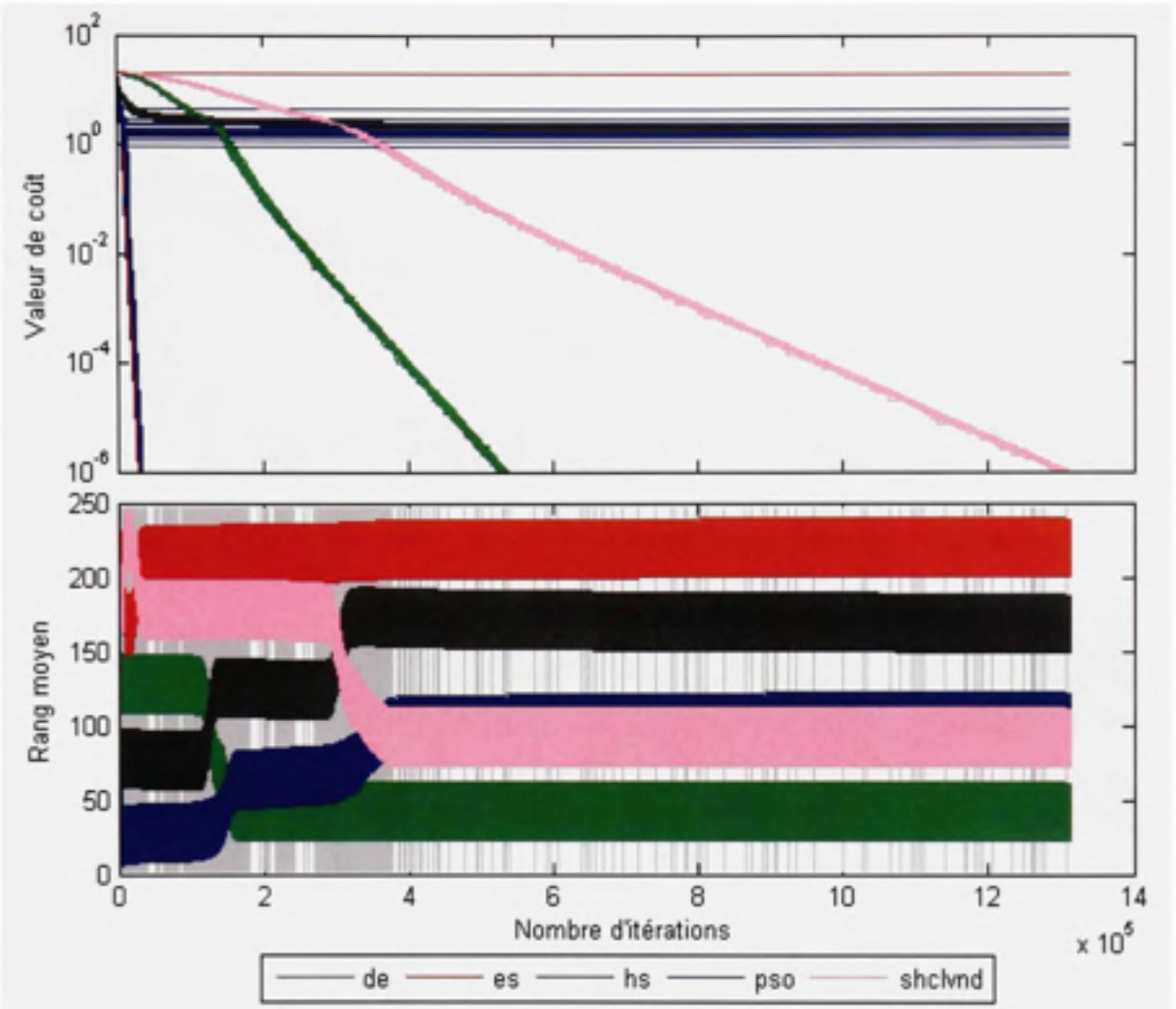

Figure V.14 ***Échantillons de courbes d'évolution (haut) et analyse des performances (bas) de la fonction Ackley en 30 dimensions.***

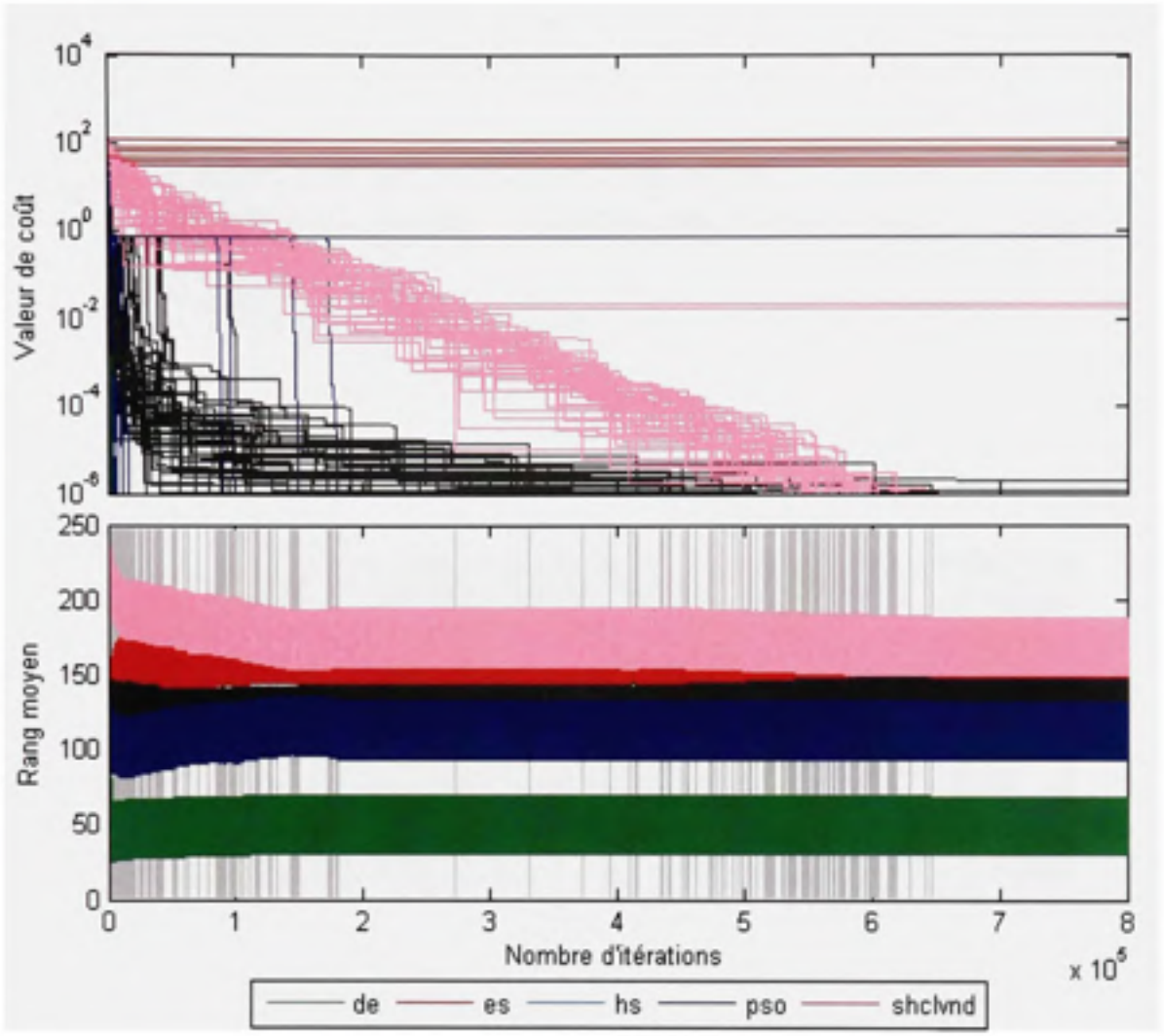

Figure V.15 ***Échantillons de courbes d'évolution (haut) et analyse des performances (bas) de la fonction Shubert pénalisée en 2 dimensions.***

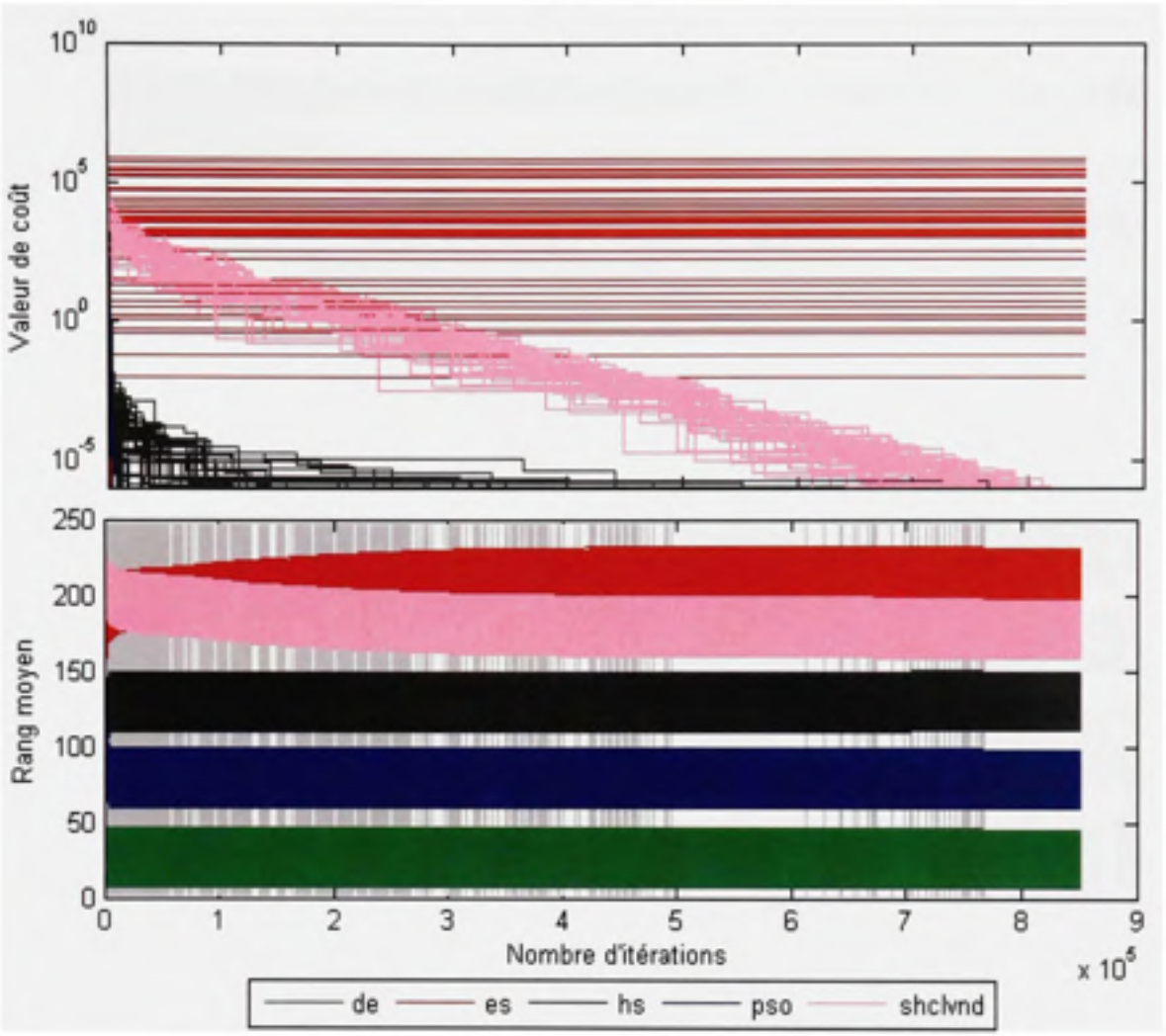

Figure V.16 ***Échantillons de courbes d'évolution (haut) et analyse des performances (bas) de la fonction Corana's Parabola en 2 dimensions.***

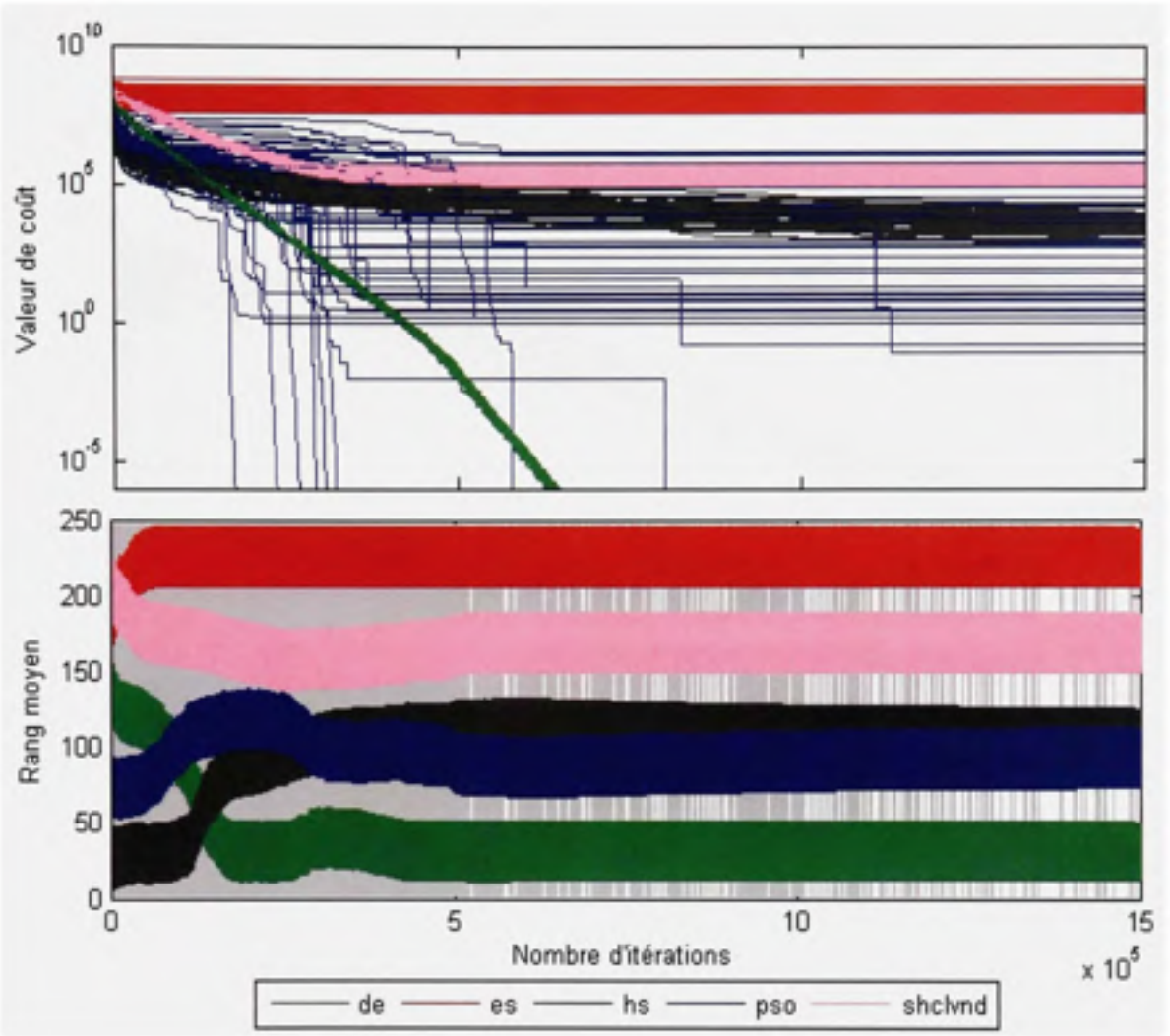

Figure V.17 ***Échantillons de courbes d'évolution (haut) et analyse des performances (bas) de la fonction Corana's Parabola en 30 dimensions.***

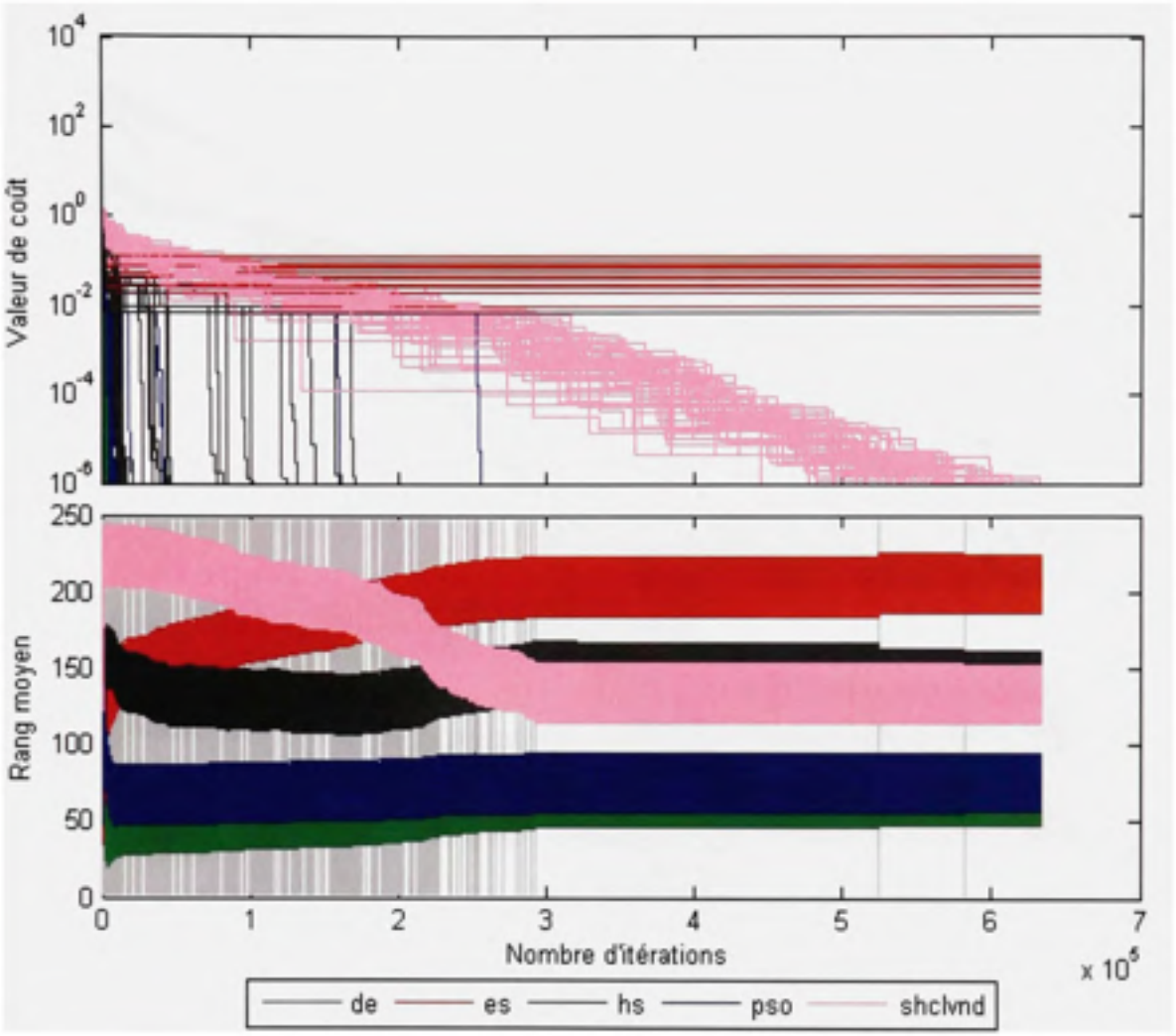

Figure V.18 ***Échantillons de courbes d'évolution (haut) et analyse des performances (bas) de la fonction Griewangk en 2 dimensions.***

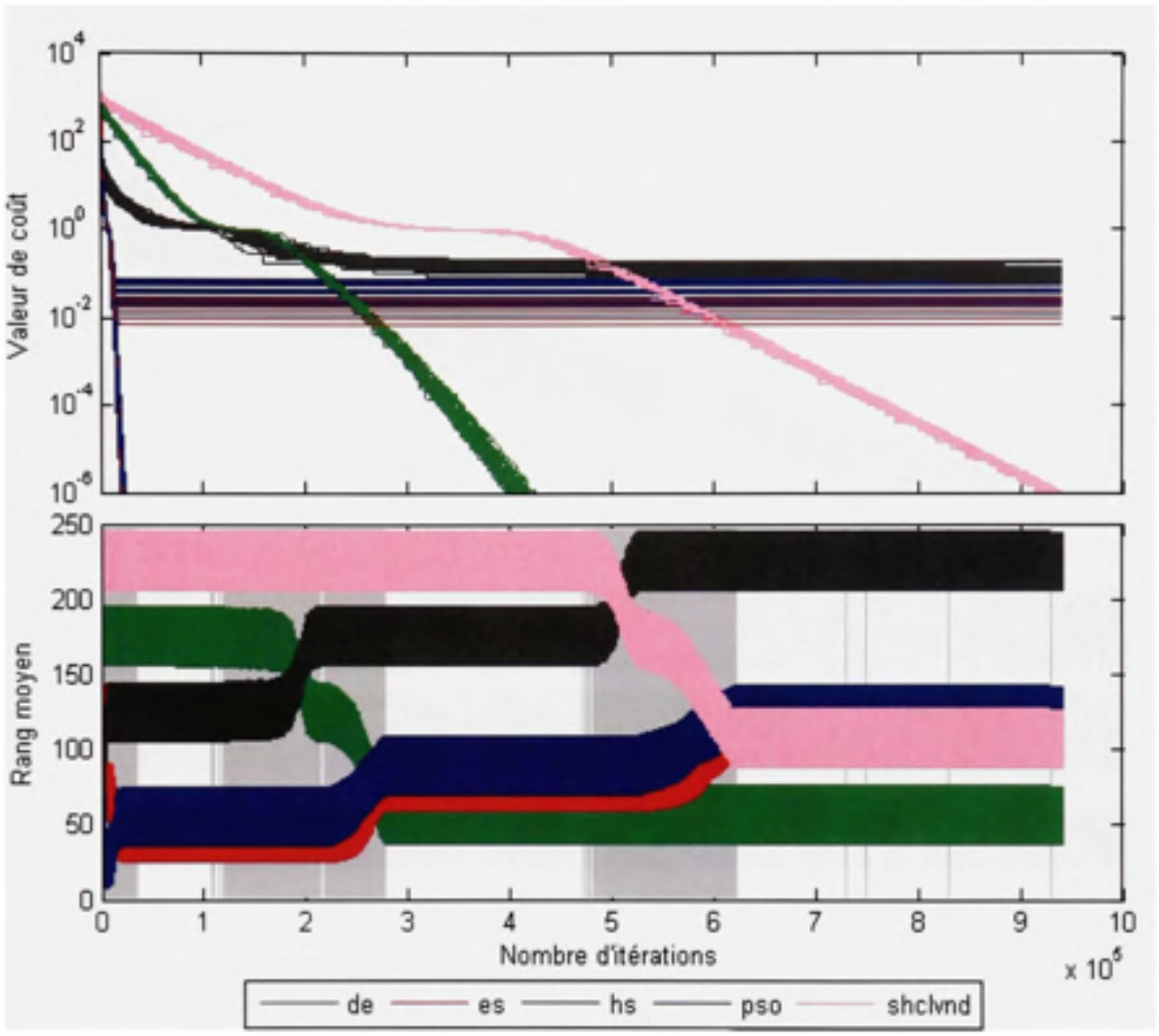

Figure V.19 ***Échantillons de courbes d'évolution (haut) et analyse des performances (bas) de la fonction Griewangk en 30 dimensions.***

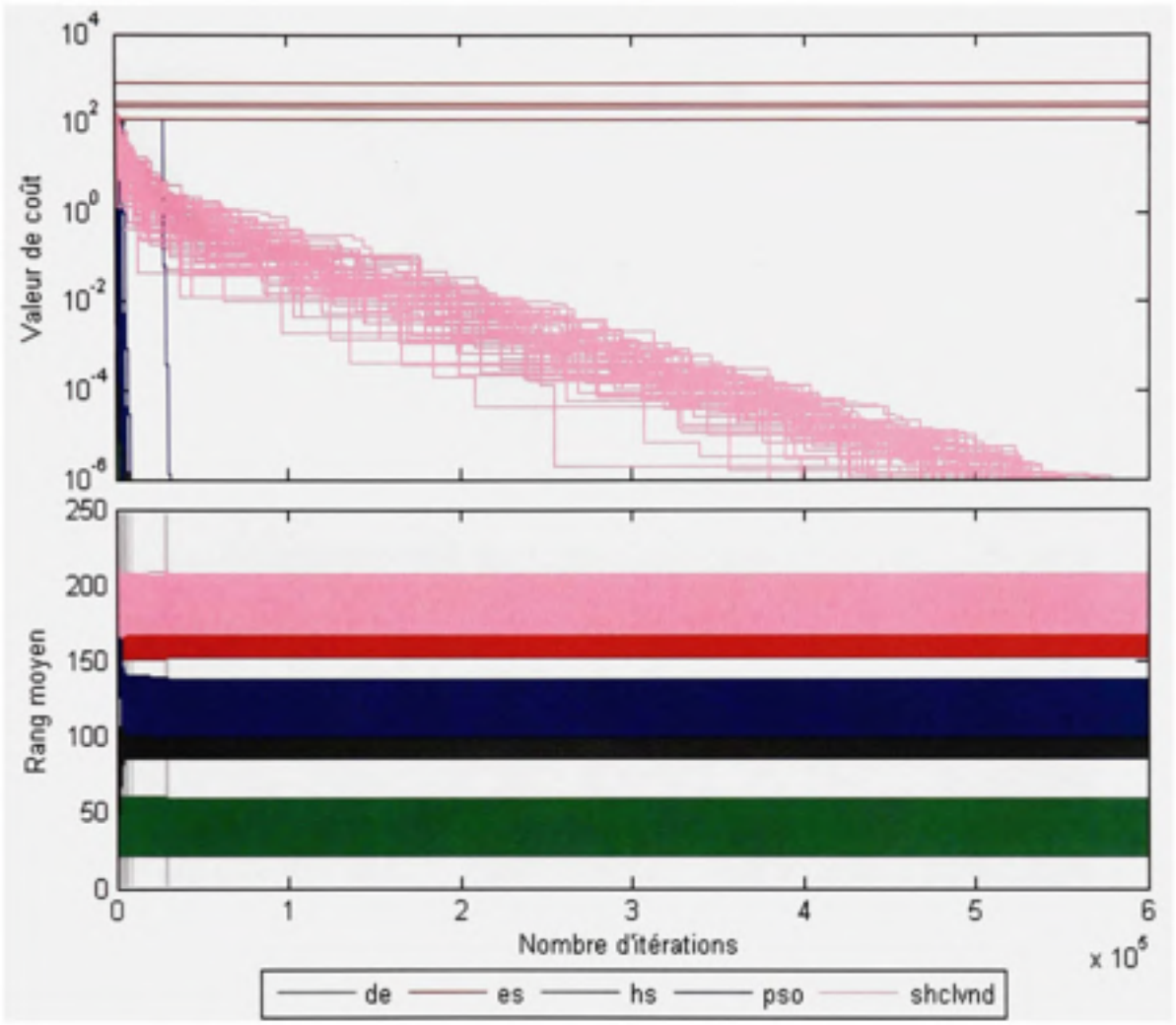

Figure V.20 ***Échantillons de courbes d'évolution (haut) et analyse des performances (bas) de la fonction Schwefel modifiée en 2 dimensions.***

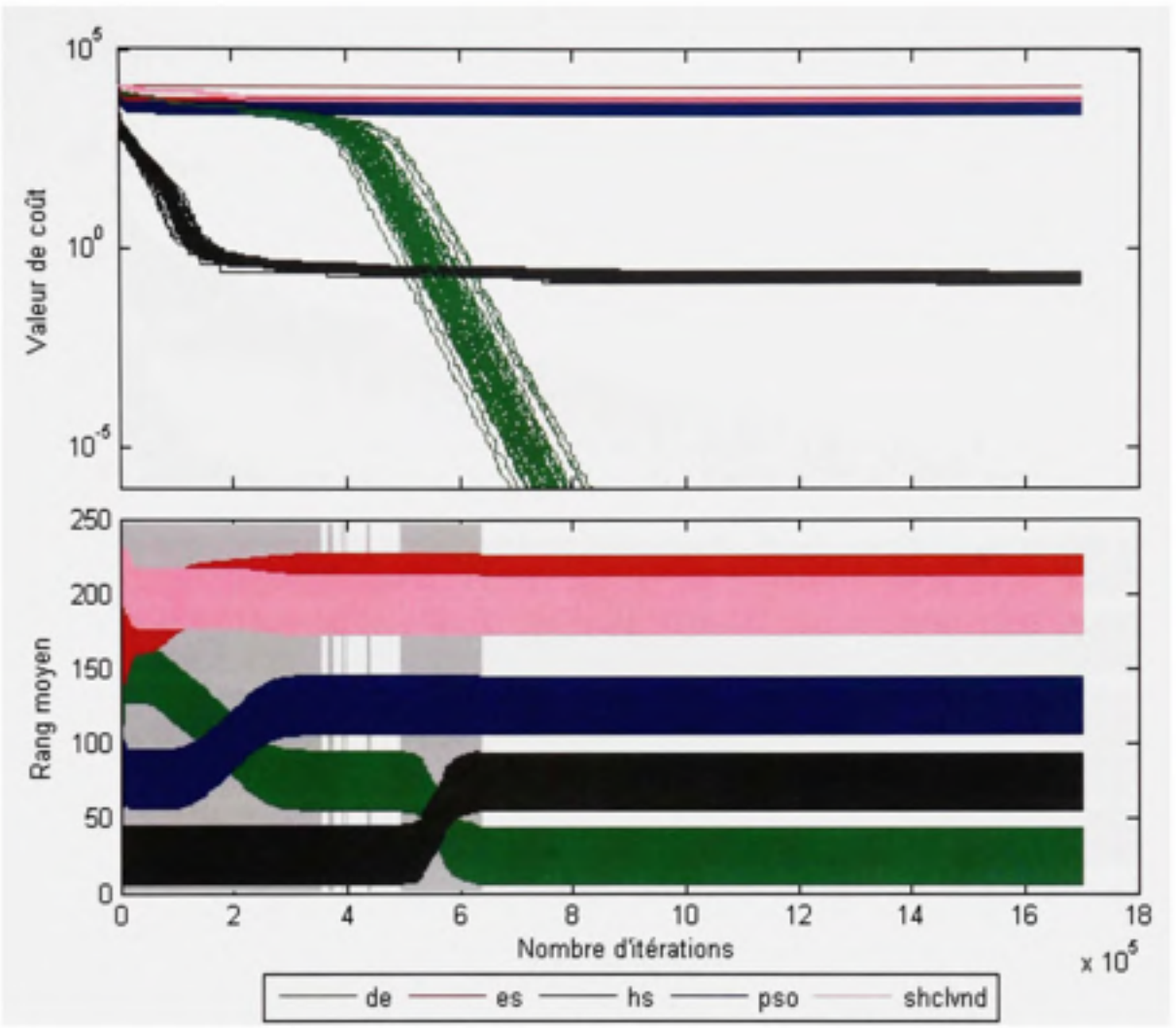

Figure V.21 ***Échantillons de courbes d'évolution (haut) et analyse des performances (bas) de la fonction Schwefel modifiée en 30 dimensions.***

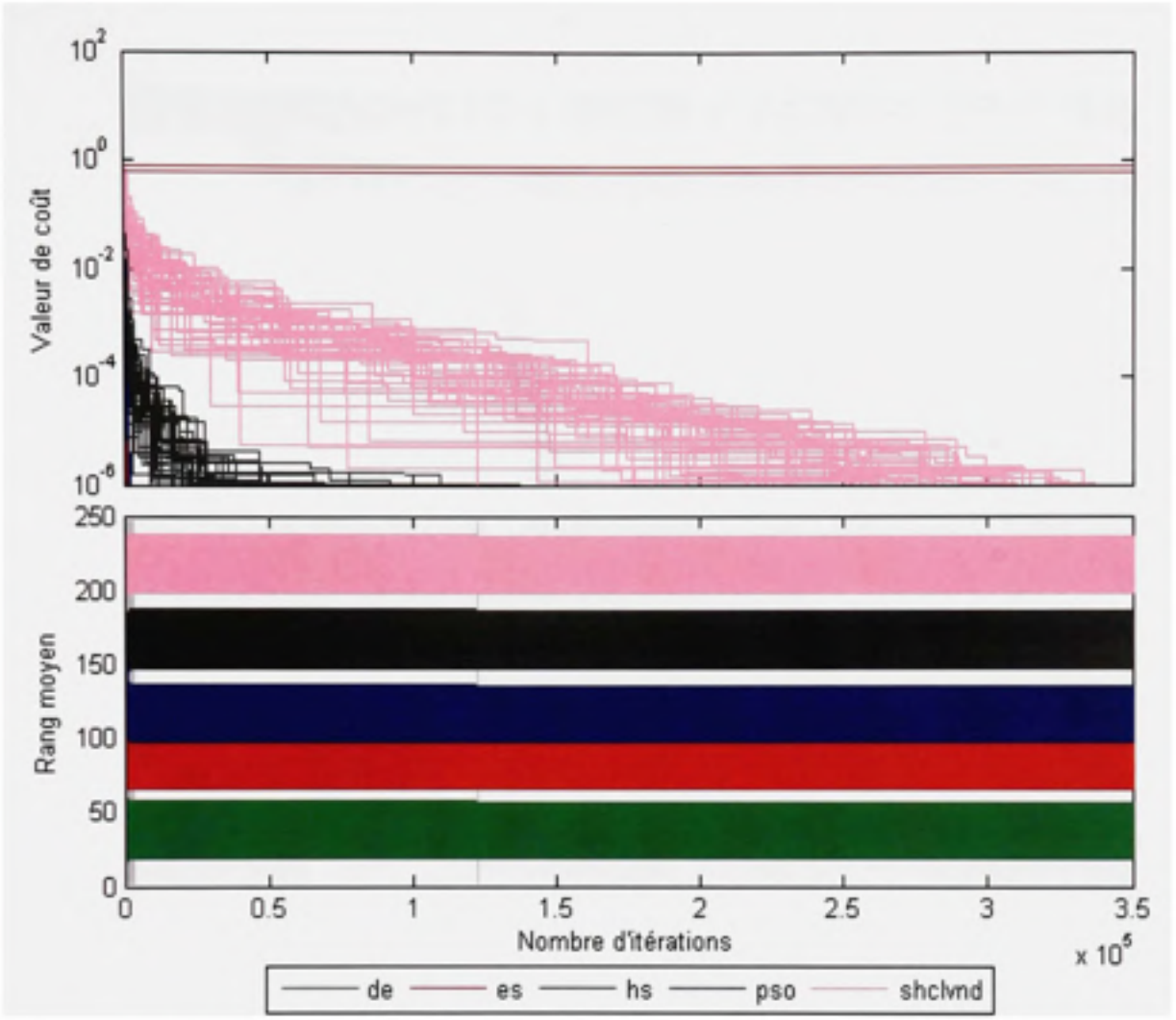

Figure V.22 ***Échantillons de courbes d'évolution (haut) et analyse des performances (bas) de la fonction Michalewicz en 2 dimensions.***

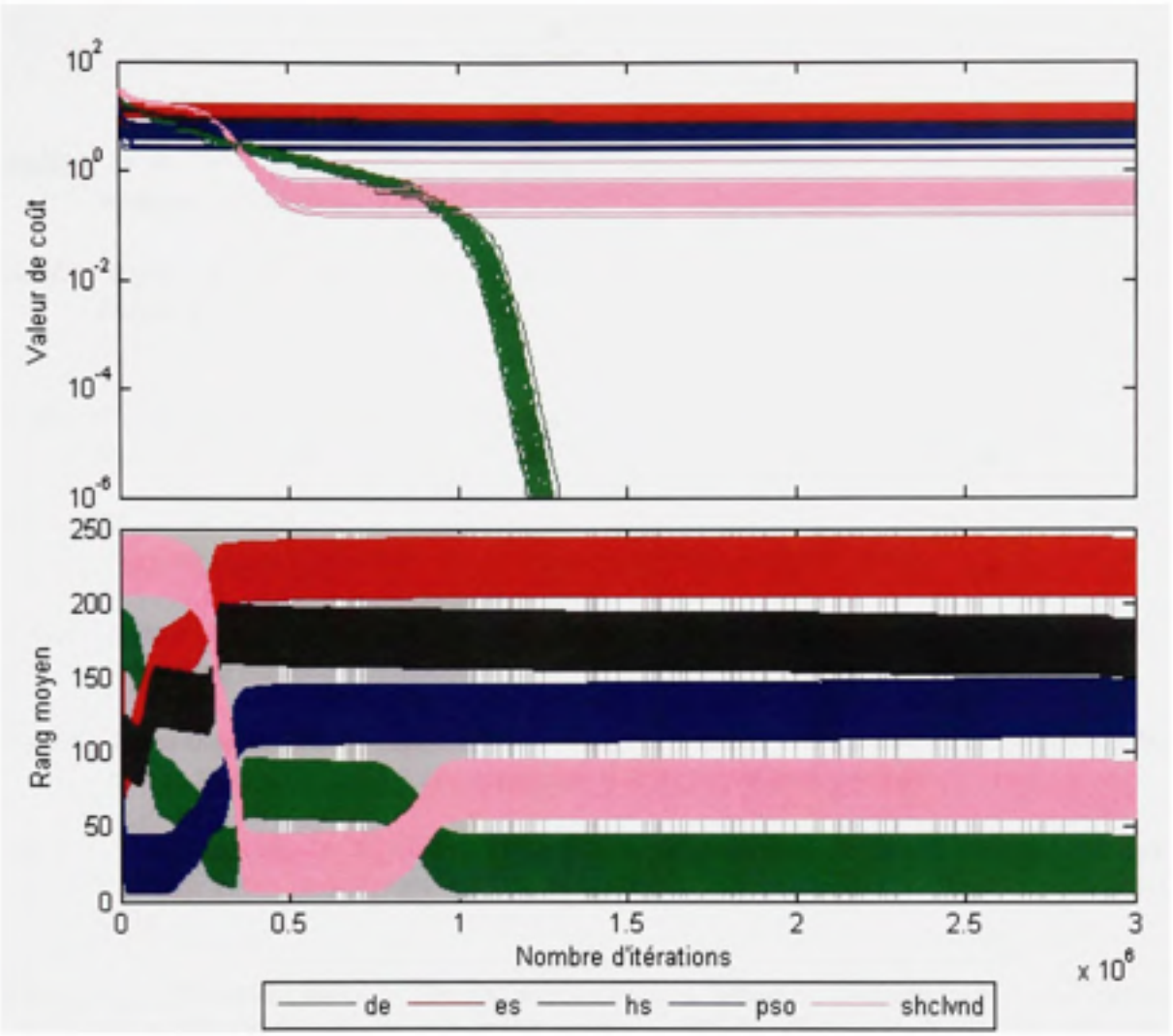

Figure V.23 ***Échantillons de courbes d'évolution (haut) et analyse des performances (bas) de la fonction Michalewicz en 30 dimensions.***

BIBLIOGRAPHIE

Ackley, D. H. 1987. *A connectionist machine for genetic hillclimbing*, 1st ed. Boston: Kluwer Academic Publishers, 260 p.

Aluffi-Pentini, F., V. Parisi, F. Zirilli. 1985. « Global Optimization and Stochastic Differential Equations ». *Journal of optimization theory and applications*, vol. 47, n° 1, p. 1-16.

Audet, Donald, Claude Boucher, André Caumartin et Claude Skeene. 1993. *Probabilités et statistiques*, Montréal : Gaëtan Morin éditeur, 410p.

Baeck, T., D.B. Fogel et Z. Michalewicz. 2003. *Evolutionary Computation 1 : Basic Algorithm and Operators*, 1st ed. Institute of Physics (IOP), 340 p.

Beyer, Hans-Georg. 2001. *The Theory of Evolution Strategies*, 1st ed. Coll. « Natural computing series ». Berlin ; New York : Springer, 380 p.

Beyer, Hans-Georg, H.-P. Schwefel. 2002. « Evolution strategies A comprehensive introduction », *Natural Computing*, vol.1, n°1, (november), p.3-52

Bhat, U. Narayan, Gregory K. Miller. 2002. *Elements of applied stochastic processes*, 3rd ed. « Wiley series in probability and statistics ». Hoboken, N.J. : Wiley-Interscience, 461p.

Clerc, M., J. Kennedy. 2002. « The Particle Swarm—Explosion, Stability, and Convergence in a Multidimensional Complex Space ». IEEE Transactions on Evolutionary Computation, vol.6, n° 1, (Februrary), p.58-73

Coello Coello, Carlos A., David A. Van Veldhuizen et Gary B. Lamont. 2002. *Evolutionary algorithms for solving multi-objective problems*, 1st ed. « Genetic algorithms and evolutionary computation », Volume 5. Boston, Mass. : Kluwer Academic, 576p.

Conover, William Jay. 1998. *Practical Nonparametric Statistics*, 3rd ed. New York, N.Y. : J. Wiley and Sons, 584p.

Corana, A., M. Marchesis, C. Martini, S. Ridella. 1987. « Minimizing Multimodal Functions of continuous Variables with the "SimulatedAnnealing Algorithm" ». *ACM Transactions on Mathematical Software*, vol. 13, n° 3, (Mars 1987), p. 272-280.

De Jong, Kenneth Alan. 1975. « An analysis of the behavior of a class of genetic adaptive systems ». Ph. D. in Computer Science, The University of Michigan, 268 p.

Dreo, J., A. Petrowski, P. Siarry et E. Taillard. 2003. *Métaheuristiques pour l'optimisation difficile*, 1ère ed. Coll. « Algorithmes ». Paris : Eyrolles, 356 p.

Eberhart, R., J. Kennedy. 1995. « A New Optimizer Using Particle Swarm Theory ». In *Proceedings of the Sixth International Symposium on Micro Machine and Human Science*. (Nagoya, Japan, October 4-6), p. 39-43

F.H., Branin. 1972. « Widely convergent method for finding multiple solutions of simultaneous nonlinear equations ». *IBM J. Res. Develop.*, vol. 16, p. 504-522.

Geem, Z.W., J.H. Kim et G.V. Loganathan. 2001. « A New Heuristic Optimization Algorithm: Harmony Search ». *Simulation*, vol.76, n°2, p. 60-68

Geem, Z.W., C.-L. Tseng. 2002. « Engineering Applications of Harmony Search ». In *Late-Breaking Papers of 2002 Genetic and Evolutionary Computation Conference (GECCO-2002)*, (New-York, July 9-13), p. 169-173.

Geem, Z.W., C.-L. Tseng. 2002. « New Methodology, Harmony Search and Its Robustness ». In *Late-Breaking Papers of 2002 Genetic and Evolutionary Computation Conference (GECCO-2002)*, (New-York, July 9-13), p. 174-178.

Gibbons Jean Dickinson. 1985. *Nonparametric Methods for Quantitative Analysis*, 2nd ed. « American series in mathematical and management sciences ». Columbus, Ohio : American Sciences Press, 481p.

Gibbons, Jean Dickinson et Subhabrata Chakraborti. 1992. *Nonparametric Statistical Inference*, 3rd ed. « Statistics, textbooks and monographs ». New York : M. Dekker, 544p.

Goldstein, A. A., I. F Price. 1971. « On descent from local minima ». *Mathematics of Computation*, Vol. 25, n° 115, p. 569-574.

Heppner, F., U. Grenander. 1990. « A stochastic nonlinear model for coordinated bird flocks ». *S. Krasner, Ed., The Ubiquity of Chaos. AAAS Publications*, p. 233-238

Ho, Y.C., D.L. Pepyne. 2002. « Simple Explanation of the No-Free-Lunch Theorem and Its Implications ». *Journal of Optimization Theory and Applications*, vol.115, n°3, (december) p. 549-570

Hochberg, Y. et A. Tamhane. 1987. *Multiple Comparison Procedures*, 1st ed. États-Unis : Wiley, 450p.

Kao Edward PC. 1997. *An introduction to stochastic processes*, 1st ed. Belmont, Calif. : Duxbury Press, 438p.

Kennedy, J., R. Eberhart. 1995. « Particle Swarm Optimization ». In *Proceedings of IEEE International Conference on Neural Networks, 1995.* (Perth (WA), Austria, Nov/Dec), p.1942-1948

Kennedy, J. 1999. « Small worlds and mega-minds: effects of neighborhood topology onparticle swarm performance ». In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 1999).* (Washington D.C., 6-9 July), vol. 3, p.1931-1938.

Kennedy, J. 2003. « Bare Bones Particle Swarms ». In *Proceedings of the 2003 IEEE Swarm Intelligence Symposium.* (Indianapolis, IN, USA, April 24-26), p.80-87.

Klockgether. J., H.P. Schwefel. 1970. « Two-Phase Nozzle And Hollow Core Jet Experiments ». In *11th Symposium on Engineering Aspects of Magneto-Hydrodynamics* (1970), p. 141-148.

Lee, K.S., Z.W. Geem. 2004. « A new structural optimization method based on the harmony search algorithm ». *Computers & Structures*, vol.82, (april), p. 781-798

Macready, W.G., et D.H. Wolpert. 1995. *What Makes an Optimization Problem Hard?*, Coll. « Santa Fe Institute working papers », SFI-TR-95-05-046. Santa Fe (NM) : Santa Fe Institute, 11p.

Price, Kenneth V., Rainer M. Storn et Jouni A. Lampinen. 2005. *Differential Evolution: A Practical Approach to Global Optimization*, 1st ed. « Natural Computing Series ». Secaucus, NJ: Springer-Verlag New York, Inc., 538 p.

Rechenberg, I. 1965. *Cybernetic solution path of an experimental problem*, Royal Aircraft Establishment, Library Translation No. 1122, August.

Rechenberg, I. 1971. « Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution ». Dr.-Ing. Thesis, Technical University of Berlin, Department of Process Engineering

Rechenberg, I. 1973. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*, Frommann-Holzboog Verlag, Stuttgart

Ross, Sheldon M. 2002. *Probability models for computer science*, 1st ed. San Diego, Calif. : Harcourt/Academic Press, 288p.

Rudolf, S., M. Köppen. 1996. *Stochastic Hill Climbing with Learning by Vectors of Normal Distributions*, (IPK), Berlin : Fraunhofer-Institut for Production Systems and Design Technology, 11 p.

Shi, Y., R. Eberhart. 1998. « A modified particle swarm optimizer ». In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 1998)*. (Piscataway, NJ, 4-9 May), p.69-73.

Shi, Y., R.C. Eberhart. 1999. « Empirical Study of Particle Swarm Optimization ». In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 1999)*. (Washington D.C., 6-9 July), vol. 3, p.1945-1950.

Storn, R., K. Price. 1995. *Differential Evolution - a Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces*. TR-95-012. Berkeley (CA) : International Computer Science Institute (ICSI), 12 p.

Storn, R., K. Price. 1997. « Differential Evolution A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces ». *Journal of Global Optimization*, vol. 11, n°4, (december), p. 341-359

Storn, Rainer. « Differential Evolution (DE) ». En ligne. <http://www.icsi.berkeley.edu/~storn/code.html>. Consulté le 25 mai 2007.

Sugant Han, P.N. 1999. « Particle Swarm Optimiser with Neighbourhood Operator ». In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 1999)*. (Washington D.C., 6-9 July), vol. 3, p.1958-1962.

Weinberg, B., E.-G. Talbi. 2004. « NFL theorem is unusable on structured classes of problems ». In *Proceedings of the 2004 Congress on Evolutionary Computation (CEC'2004)*. Vol.1. (Portland OR, USA, june 19-23 2004), p.220-226.

Wilson, Edward O. 1975. *Sociobiology: The new synthesis*, 1st ed. Cambridge, Mass: Belknap Press of Harvard University Press, 697 p.

Wolpert, D.H., W.G. Macready. 1997. « No Free Lunch Theorems for Optimization ». *IEEE Transactions on Evolutionary Computation*, vol. 1, n°1, (avril) p. 67-82.