Variations sur PSO : approches parallèles, jeux de voisinages et applications

Maria Zemzami

Normandie Université

THESE EN COTUTELLE INTERNATIONALE

Pour obtenir le diplôme de doctorat

Spécialité Informatique

Préparée au sein de L'INSA Rouen Normandie

VARIATIONS SUR PSO : APPROCHES PARALLELES, JEUX DE VOISINAGES ET APPLICATIONS.

Présentée et soutenue par

Maria ZEMZAMI

Thèse soutenue publiquement le 06 Juillet 2019 devant le jury composé de		
Mr Abdellah El MOUDNI	PU EX1 Université de Technologie Belfort-Montbéliard	Examinateur
Mr Mhamed ITMI	MCF HDR INSA Rouen Normandie	Directeur de thèse
Mr Nabil HMINA	PES Président Université Sultan Moulay Slimane. Maroc	Codirecteur de thèse
Mr Norelislam ELHAMI	PH ENSA Université Ibn Tofail Kenitra. Maroc	Encadrant
Mr Yann POLLET	PU EX1 responsable chaire informatique CNAM Paris	Rapporteur
Mme Hanaa HACHIMI	PH Université Sultan Moulay Slimane. Maroc	Rapporteur
Mr Bouchaib RADI	PES Université Hassan 1er. Settat. Maroc	Rapporteur
Mr Abdelkhalak ELHAMI	PU EX1 INSA Rouen Normandie	Examinateur

Thèse dirigée par Nabil HMINA et Mhamed ITMI, laboratoires LGS & LITIS respectivement.

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES ROUEN

MIIS

Résumé

Reconnue depuis de nombreuses années comme une méthode efficace pour la résolution de problèmes difficiles, la méta-heuristique d'optimisation par essaim de particules PSO (Particle Swarm Optimization) présente toutefois des inconvénients dont les plus étudiés sont le temps de calcul élevé et la convergence prématurée.

Cette thèse met en exergue quelques variantes de la méthode PSO visant à échapper à ces deux inconvénients de la méthode. Ces variantes combinent deux approches : la parallélisation de la méthode de calcul et l'organisation de voisinages appropriés pour les particules. L'évaluation de la performance des modèles proposés a été effectuée sur la base d'une expérimentation sur une série de fonctions tests. A la lumière de l'analyse des résultats expérimentaux obtenus, nous observons que les différents modèles proposés donnent des résultats meilleurs que ceux du PSO classique en termes de qualité de la solution et du temps de calcul.

Un modèle basé PSO a été retenu et développé en vue d'une expérimentation sur le problème du transport d'électricité. Une variante hybride de ce modèle avec la méthode du recuit simulé SA (Simulated Annealing) a été considérée et expérimentée sur la problématique des réseaux de collaboration.

Mots clés. Optimisation, optimisation par essaim de particules, parallélisation, recuit simulé, réseaux de collaboration, transport d'électricité, voisinage.

Abstract

Known for many years as a stochastic metaheuristic effective in the resolution of difficult optimization problems, the Particle Swarm Optimization (PSO) method, however, shows some drawbacks, the most studied: high running time and premature convergence.

In this thesis we consider some variants of the PSO method to escape these two disadvantages. These variants combine two approaches: the parallelization of the calculation and the organization of appropriate neighborhoods for the particles. To prove the performance of the proposed models, we performed an experiment on a series of test functions. By analyzing the obtained experimental results, we observe that the proposed models based on the PSO algorithm performed much better than basic PSO in terms of computing time and solution quality.

A model based on the PSO algorithm was selected and developed for an experiment on the problem of electricity transmission. A hybrid variant of this model with Simulated Annealing (SA) algorithm has been considered and tested on the problem of collaborative networks.

Key words. Collaborative networks, electricity transmission, neighborhood, Optimization, parallelism, Particle Swarm Optimization, Simulated Annealing.

ملخص

خوارزمية التحسين باستعمال سرب من الجسيمات هي إحدى الطرق المعروفة في مجال التحسين منذ سنوات عديدة و فعالة في حل مشكلات التحسين الصعبة .
ورغم فعاليتها فهي كباقي طرق التحسين تظهر بعض السلبيات، من بين الأكثر دراسة : ارتفاع إدارة الوقت والتقارب السابق لأوانه.

في هذه الأطروحة نقترح مشاركات علمية كحلول لتفادي هذه السلبيات، عن طريق إنشاء نماذج تستند على طريقة خوارزمية التحسين باستعمال سرب من الجسيمات باستخدام الحوسبة المتوازية ومفهوم الأسراب الفرعية.

بغرض تجريب كفاءة وفعالية النماذج المقترحة ، أجرينا تجربة على سلسلة من وظائف الاختبار من خلال تحليل النتائج التجريبية التي تم الحصول عليها، نلاحظ أن النماذج المقترحة المستندة على طريقة خوارزمية التحسين باستعمال سرب من الجسيمات كانت أفضل بكثير من حيث وقت الحساب وجودة الحل .
تم اختيار وتطوير نموذج يعتمد على طريقة خوارزمية التحسين باستعمال سرب من الجسيمات لإجراء تجربة على مشكلة تحسين نقل الكهرباء ومن ثم تم تطويرها واختبارها على مشكلة الشبكات التعاونية.

الكلمات المفتاحية : الأسراب الفرعية، التحسين ، التحسين باستعمال سرب من الجسيمات، الشبكات التعاونية الحوسبة المتوازية، نقل الكهرباء .

Remerciements

Je tiens à exprimer ma plus sincère reconnaissance à Monsieur **Nabil HMINA**, Président de l'Université Sultan Moulay Slimane de Béni mellal, sans qui cette thèse n'aurait sûrement pas eu lieu. En effet, il a sans conteste été la personne qui, grâce à son dynamisme et à son enseignement remarquable, a su me donner ce goût véritable pour les travaux de thèse tout en renforçant sans cesse mes liens avec la recherche scientifique. Il m'a immédiatement accordé sa confiance et m'a constamment encouragé, conseillé, guidé avec une gentillesse et une disponibilité qui lui sont propres. Depuis plusieurs années, cela a toujours été un grand honneur et un véritable plaisir de travailler ou tout simplement de discuter avec une personne possédant de telles qualités scientifiques et humaines. Sans ses encouragements dans les phases de doute, sans son aide constante et la pertinence de ses conseils, cette thèse ne serait pas ce qu'elle est.

J'exprime ici ma profonde gratitude à Monsieur **Mhamed ITMI**, MCF.HDR à l'INSA de Rouen, Université de Normandie. Ses conseils et ses encouragements ont permis à ce travail d'aboutir. Ses capacités scientifiques et ses compétences étaient mon grand support. Merci infiniment à vous pour votre patience et la grande confiance que vous m'avez accordée. Merci pour toutes les discussions, les idées et les remarques (toujours pertinantes et constructives) qui font avancer.

Je remercie vivement Monsieur **Norelislam El HAMI,** Proofesseur Habilité à l'ENSA de Kenitra, Université Ibn Tofail, pour l'aide technique qu'il m'a portée au cours de ma thèse, pour sa disponibilité, son écoute, sa réactivité, ses conseils qui m'ont été d'une grande utilité, et surtout pour tous les efforts et le temps qu'il a consacré pour améliorer mon manuscrit.

Je tiens à t'émoigner toute ma sympathie à Monsieur **Abdelkhalak El HAMI,** PU EX1, à l'INSA de Rouen, Université de Normandie. Il fut toujours d'une aide scientifique et

psycologique inestimable en faisant preuve d'une incroyable disponibilité et gentillesse sans faille. Il s'est constamment intéressé à mon travail, et m'a toujours écouté et m'a guidé avec patience et compétence.

J'exprime ma profonde gratitude à Monsieur **Abdellah El MOUDNI,** PU EX1 à l'Université de Technologie de Belfort-Montbéliard en France, pour m'avoir fait l'honneur de présider mon jury de thèse. Je lui adresse par ailleurs toute ma reconnaissance pour avoir accepté de se déplacer au Maroc pour présider ma soutenance.

Mes sincères remerciements vont aussi à Madame **Hanaa HACHIMI,** Proofesseur Habilité à l'Université Sultan Moulay Slimane de Béni mellal, Monsieur **Yan POLLET,** PU EX1 responsable chaire informatique Cnam Paris, et Monsieur **Bouchaib RADI**, PES à l'université Hassan 1er de Settat, pour avoir, malgré leurs lourdes charges, accepté de rapporter sur ma thèse.

Je voudrais profiter de ce manuscrit pour remercier tous les membres du laboratoire LITIS pour leur accueil et leur soutien durant mes séjours à l'INSA.

Je ne saurais oublier mes parents qui m'ont apporté l'éducation nécessaire à ce travail et m'ont toujours donné la force et le courage indispensables pour progresser. C'est à ma mère que je dois toute ma rigueur dans le travail et surtout mon amour pour l'apprentissage. Je remercie vivement mon père pour tous ses conseils et aussi pour son soutien moral qui fut pour moi la meilleure motivation. Je n'oublie pas également ma soeur ainée Sanae et mon petit frère Mouad qui avec une profonde gentillesse ont constamment été présents pour m'encourager et me soutenir. Enfin, je voudrais que mon mari qui m'as toujours apporté soleil et gaieté mais aussi le courage lorque j'étais lasse et l'espoir lorsque je n'en n'avais plus, qu'il trouve ici bien autre chose que ma gratitude.

Je dédie cette thèse
À mes parents,
À ma soeur,
À mon frère,
À mon mari,
À mes petites princesses Amira et Mira,
À mes grands-parents,
À toute ma famille.
Et plus particulièrement, à ma tante Mina,
Ainsi qu'à tous mes amis(es).

Table des matières

Liste de mes publications

Liste des articles publiés :

- Maria ZEMZAMI, Norelislam ELHAMI, Abderahman MAKHLOUFI, Mhamed ITMI et Nabil HMINA. « Electrical Power Transmission Optimization based on a New Version of PSO Algorithm ». (Publié le 22/02/17 DOI: 10.21494/ISTE.OP.2017.0127).
- Maria ZEMZAMI, Norelislam ELHAMI, Mhamed ITMI et Nabil HMINA. « A modified Particle Swarm Optimization algorithm linking dynamic neighborhood topology to parallel computation ». International Journal of Engineering and Technology (IJATCSE) Volume 8, N2 pp 112-118 (2019).
- Maria ZEMZAMI, Norelislam ELHAMI, Mhamed ITMI et Nabil HMINA. « An evolutionary hybrid algorithm for complex optimization problems ». International Journal of Engineering and Technology (IJATCSE) Volume 8, N 2, pp 126-133 (2019).
- Maria ZEMZAMI, Norelislam ELHAMI, Mhamed ITMI et Nabil HMINA. « Interoperability optimization using a modified PSO algorithm ». International Journal of Engineering and Technology (IJATCSE) Volume 8, N2 pp 101-107 (2019).
- Maria ZEMZAMI, Norelislam ELHAMI, Mhamed ITMI and Nabil HMINA. « A comparative study of three parallel models based on the PSO algorithm ». International Journal for Simulation and Multidisciplinary Design Optimization (IJSMDO). (under review).
- Aicha KOULOU, Maria ZEMZAMI, Norelislam ELHAMI, et Nabil HMINA « Optimization in collaborative information systems for an enhanced interoperability

network ». International Journal for Simulation and Multidisciplinary Design Optimization (IJSMDO). (under review).

Communications dans des conférences internationales :

- Maria ZEMZAMI, Norelislam ELHAMI, Mhamed ITMI et Nabil HMINA. « Parallèlisation de la Méthode PSO: découpage de l'espace et traitement par lot des particules ». 1st International Workshop on New Services and Networks (WNSN'2016). 23-24 Mai 2016 Khouribga, Maroc.
- Maria ZEMZAMI, Norelislam ELHAMI, Mhamed ITMI et Nabil HMINA. « Parallèlisation de la méthode PSO sur des voisinages évolutifs ». 11th International Conference on Modeling, Optimization and Simulation (MOSIM'16). 22-24 Aout 2016 Montréal, Canada.
- Maria ZEMZAMI, Norelislam ELHAMI, Mhamed ITMI et Nabil HMINA. « A new parallel approach for the exploitation of the search space based on PSO algorithm ». 4th International Colloquium on Information Science and Technology (IEEE-CIST'16). 24-26 October 2016 Tangier, Maroc. (Indexée Scopus)
- Maria ZEMZAMI, Norelislam ELHAMI, Mhamed ITMI et Nabil HMINA. « Modified Particle Swarm Optimization algorithm based on a dynamic search space». 3rd International Congress on Advanced Technologies (IEEE-ICAT'17). 12-14 Avril 2017 Safi, Maroc.
- Maria ZEMZAMI, Abdelkhalak ELHAMI, Abderahman MAKHLOUFI, Norelislam ELHAMI, Mhamed ITMI et Nabil HMINA. « Applying a new parallelized version of PSO algorithm for electrical power transmission ». International Conference on Materials Engineering and Nanotechnology (ICMEN'17). 12-14 Mai 2017 Kuala Lumpur, Malysia**.** IOP Conference Series: Materials Science and Engineering (Online ISSN: 1757-899X; Print ISSN: 1757-8981) indexée: EI Compendex, Scopus, Thomson Reuters (WoS), Inspec,et al.

- Maria ZEMZAMI, Norelislam ELHAMI, Mhamed ITMI et Nabil HMINA. « A hybrid algorithm based on Particle Swarm Optimization and Simulated Annealing for electrical power transmission ». The 7th International Conference on Metaheuristics and Nature Inspired Computing (META'18). 27-31 Octobre 2018. Marrakech, Maroc.
- Maria ZEMZAMI, Norelislam ELHAMI, Mhamed ITMI et Nabil HMINA. « An efficient hybrid model based on Particle Swarm Optimization and Simulated Annealing algorithms for complex optimization problems ». 6th International Conference on Computer Science, Engineering and Technologies (ICCSET'18). 3-4 Septembre 2018. Bangkok, Thailand.
- Maria ZEMZAMI, Norelislam ELHAMI, Mhamed ITMI et Nabil HMINA. « A Novel Particle Swarm Optimization Algorithm linking Dynamic Neighborhood Topology to Parallel Computation for Complex Optimization Problems ». 4th International Conference on Computing, Engineering & Emerging Technologies (ICCEET'18). 1-2 Octobre 2018. Dubai, UAE.
- Maria ZEMZAMI, Norelislam ELHAMI, Mhamed ITMI et Nabil HMINA. « Optimizing Interoperability using a new parallel model based on the PSO algorithm ». 5th International Conference on Computing, Engineering & Emerging Technologies (ICCEET'18). 18-19 Octobre 2018. Singapour.

Liste des abréviations

APDL : ANSYS Parametric Design Language.

CPU: Central Processing Unit.

CSP : Constraint Satisfaction Problems.

CUDA : Compute Unified Device Architecture.

ENSA : École Nationale des Sciences Appliquées.

FPGA : Field Programmable Gate Array.

GPU : Graphics Processing Unit.

IHM : Interface Homme Machine.

INSA : Institut National des Sciences Appliquées.

LGS : Laboratoire des Génies et Systèmes.

LITIS : Laboratoire d'Informatique et Traitement de l'Information et des Systèmes.

MPI : Message-Passing Interface.

MPSO : Modified Particle Swarm Optimization.

OpenCL : Open computing language.

OpenGL : Open graphics library.

OpenMP : Open multiprocessing.

PSO : Particle Swarm Optimization.

PVM : Parallel virtual machine.

PPSO : Parallel Particle Swarm Optimization.

SA : Simulated Annealing.

SI : Système d'Information.

SR : Success Rate.

SD : Standard Deviation.

TPU: Tensor Processing Unit.

Liste des tableaux

Liste des figures

Liste des figures Annexe

Introduction Générale

Cette thèse en cotutelle s'inscrit dans le cadre de la collaboration entre le Laboratoire des Génies et Systèmes (LGS, École Nationale des Sciences Appliquées, Université Ibn Tofail, à Kenitra au Maroc) et le Laboratoire d'Informatique et Traitement de l'Information et des Systèmes (LITIS, Institut National des Sciences Appliquées, Université de Normandie à Rouen en France).

Les problèmes d'optimisation dans le monde réel sont généralement complexes. Ils contiennent non seulement les termes de contraintes, d'objectifs simples / multiples, mais leur modélisation évolue constamment. Leur résolution et l'évaluation itérative des fonctions objectives requièrent un temps CPU long. Le coût de calcul élevé pour la résolution de ces problèmes a poussé au développement d'algorithmes d'optimisation avec la parallélisation.

L'algorithme d'optimisation par essaims de particules (PSO) est l'un des algorithmes les plus populaires basés sur l'intelligence des essaims, qui s'enrichit de robustesse, de simplicité et de capacités de recherche globale. Cependant, l'un des principaux obstacles de la méthode PSO est sa susceptibilité à rester piégé dans les optima locaux et; à l'instar d'autres algorithmes évolutifs, les performances de PSO se détériorent dès que la dimension du problème augmente. Par conséquent, plusieurs efforts sont déployés pour améliorer ses performances. Différents scénarios sont développés, soit en ajoutant de nouveaux paramètres à l'algorithme de base [1], en l'hybridant avec d'autres méta-heuristiques [2], ou en proposant des scénarios de parallélisation [3] - [6].

L'architecture de base de PSO hérite d'un parallélisme naturel et la réceptivité des machines de traitement rapide a rendu cette tâche très pratique. Par conséquent, les modèles parallèles basés sur la méta-heuristique PSO sont devenus très populaires car la parallélisation propose un excellent chemin à l'amélioration des performances du système.

L'objectif de ce travail de thèse est la proposition de modèles basés sur la méta-heuristique PSO qui visent à améliorer les performances de cette dernière en terme de qualité de la solution et du temps de calcul. On considère diverses variantes de la méthode PSO en combinant deux concepts : la parallélisation et le voisinage scrutés sous divers ongles, dans le but d'échapper aux deux inconvénients de la méthode précédemment cités : la convergence prématurée et le temps élevé de calcul.

Organisation du document

Ce rapport de thèse se présente comme suit :

Le premier chapitre aborde un état de l'art des différents concepts utilisés dans ce travail de thèse, principalement le concept du parallélisme qui est d'une importance capitale dans la réalisation des différents modèles parallèles présentés dans ce manuscrit. Une partie a été consacrée à la notion de l'optimisation en se focalisant sur les méthodes approchées méta-heuristiques, un intérêt particulier a été porté à la méthode PSO qui fait l'objet principal de ce travail de thèse. Une description de la méthode SA a été proposée à la fin de ce chapitre.

Le deuxième chapitre amorce la présentation d'une séléction d'algorithmes parallèles portant sur la méta-heuristique PSO. Nous proposons ensuite deux modèles PPSO1 et PPSO2 basés sur l'algorithme PSO et visant à échapper aux deux inconvénients de la méthode : la convergence prématurée et le temps de calcul élevé. Une description détaillée des deux modèles proposés ainsi que leurs expérimentations pratiques sur une série de fonctions tests ont été développées.

Dans le troisième chapitre un modèle basé PSO est proposé en utilisant les threads pour le calcul parallèle, ainsi qu'une notion de voisinages dynamiques pour éviter la convergence prématurée de la méthode; ce modèle nommé MPSO a été appliqué sur la problématique du transport d'électricité. Plus précisément l'optimisation de la durée de vie du pylône d'une ligne de transport d'électricité. L'objectif été de maximiser la résistance à la charge tout en réduisant le coût « minimisation de l'utilisation des matériaux ».
Dans nos expérimentations, les tests effectués sur le programme ont donné des résultats satisfaisants du modèle parallèle proposé par rapport au processus d'optimisation de premier ordre d'ANSYS.

Une variante nommée H-MPSO-SA du modèle MPSO a été développée dans **le quatrième chapitre** de ce manuscrit. Elle est basée sur l'hybridation du modèle MPSO avec l'algorithme SA dans le but d'améliorer la qualité des résultats obtenus du modèle MPSO. H-MPSO-SA a été testée en premier lieu sur ensemble de fonctions test. Les réusltats obtenus ont montré une amélioration de la qualité de la solution obtenue. Ensuite H-MPSO-SA a été appliquée à la problématique des réseaux de collaboration.

Nous concluons cette recherche par une synthèse des différentes contributions. Nous proposons ensuite des évolutions qui devraient rendre notre modèle basé sur la méta-heuristique PSO encore plus efficace.

Chapitre 1 Principes du parallélisme et méthodes métaheuristiques

SOMMAIRE

1.1 Introduction

La plupart des problèmes de recherche complexes peuvent être formulés comme des problèmes d'optimisation. L'émergence des grandes technologies de données a également commencé la génération de problèmes d'optimisation complexes de grande taille. Le coût de calcul élevé de ces problèmes a rendu nécessaire le développement d'algorithmes d'optimisation avec parallélisation.

Le premier chapitre fournit la motivation pour ce sujet en décrivant les étapes nécessaires pour la résolution d'un problème parallèle, ainsi qu'une classification des différents types du parallélisme. Une partie sur les différentes stratégies de la parallélisation a été développée.

Le concept du parallélisme a été d'une grande importance dans ce travail de thèse, plus précisément dans la réalisation des modèles basés sur la méthode PSO dans le but d'accélérer les calculs spécialement pour les problèmes d'optimisation complexes.

La deuxième partie de ce chapitre aborde les méthodes d'optimisation stochastiques, spécialement les méthodes approchées méta-heuristiques; en se focalisant sur la méthode d'optimisation par essaim particulaire (PSO) qui constitue le thème principal de ce travail de thèse ; cette dernière connue par sa grande performance dans le domaine de l'optimisation, elle est basée sur les interactions sociales entre les individus évoluant en essaim et contient un ensemble de paramètres qui seront décrits dans ce chapitre. Ainsi, nous abordons dans les chapitres qui suivent, dans le cas mono-objectif, son hybridation avec la méthode du recuit simulé (SA) qui est inspirée d'un processus métallurgique pour éviter les optimums locaux ; son fonctionnement et l'algorithme seront développés dans ce chapitre.

1.2 Problème du parallélisme en informatique

En programmation, le parallélisme est l'exécution simultanée de calculs (éventuellement liés) dans le but d'accélerer le traitement des problèmes de calculs intensifs et d'effectuer un grand nombre d'opérations en un temps restreint. Il englobe tous les concepts liés aux systèmes multi-processeurs / multi-coeurs, le distribué à savoir les multi-ordinateurs (qui représentent des machines composées d'un ensemble de processeurs avec mémoire distribuée interconnectés par un réseau); et aux applications associées, en passant par la conception, la modélisation, la mise en œuvre et l'utilisation des systèmes et des applications.

1.2.1 Types du parallélisme

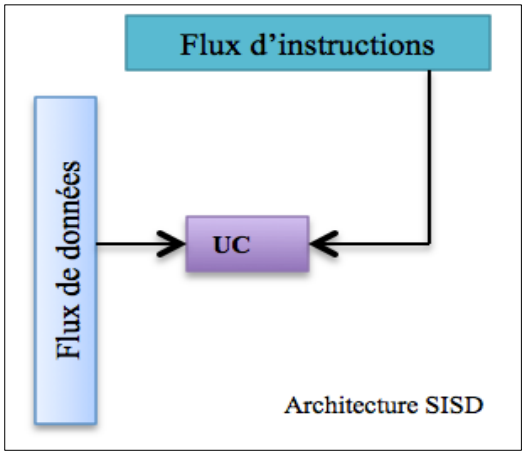
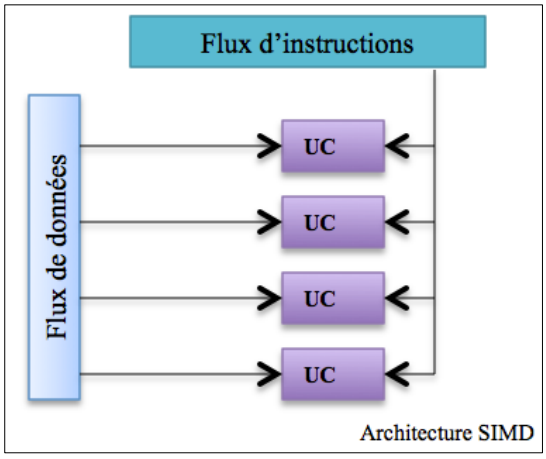
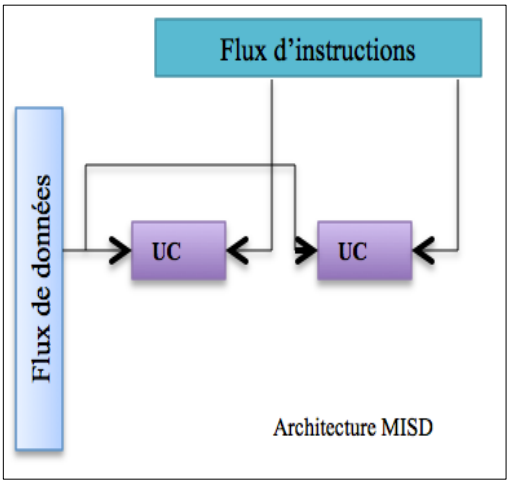
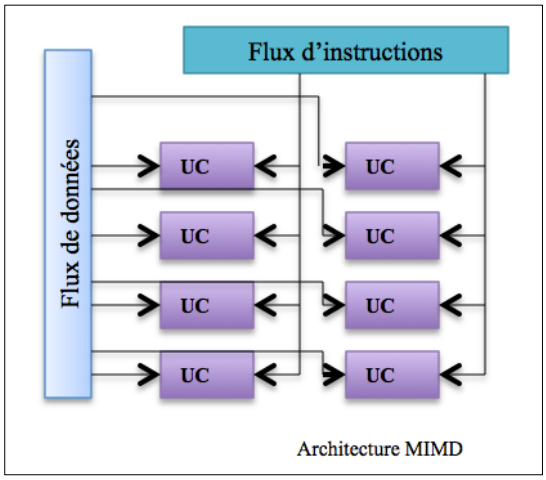
Il existe plusieurs classifications des différents types du parallélisme. Dans ce manuscrit nous allons présenter deux types de classifications : le classement de Mickeal J.Flynn nommé « la Taxonomie de Flynn » [7] et la classification par mémoire [8].

1.2.1.1 Taxonomie de Flynn

Elle représente le classement des machines selon deux dimensions indépendantes : le flux de données et le flux d'instructions. Un seul état (unique ou multiple) est attribué à chaque dimension (voir Tableau 1-1).

1) Architecture SISD (Single Instruction Single Data): il s'agit des systèmes séquentiels qui font le traitement d'une donnée à la fois.
2) Architecture SIMD (Single Instruction Multiple Data): représente des systèmes parallèles qui traitent de grandes quantités de données d'une manière uniforme.
3) Architecture MIMD (Multiple Instruction Multiple Data): ce sont des systèmes parallèles, comme les SIMD traitant de grandes quantités de données mais d'une manière hétérogène.
4) Architecture MISD (Multiple Instruction Single Data): ce sont des systèmes parallèles qui traitent une seule donnée de manière hétérogène.

Tableau 1-1 Représentation de la Taxonomie de Flynn.

	Instruction unique	**Instructions multiples**
Donnée unique	**SISD**  Exemple : Monoprocesseurs Processeurs scalaires	**SIMD**  Exemple : Processeurs vectoriels GPU
Données multiples	**MISD**  Exemple: Réseaux systoliques	**MIMD**  Exemple. : Multiprocesseurs Multi-cœurs Multi-ordinateurs

1.2.1.2 Classification par mémoire

Dans le parallélisme, nous distinguons généralement trois types de systèmes basés mémoire [8]: Les systèmes à mémoire distribuée, les systèmes à mémoire partagée et les systèmes à mémoire hybride partagée distribuée :

1. Concernant les systèmes à mémoire distribuée : Chaque processeur a sa propre portion de mémoire et ses propres données, l'espace mémoire est fragmenté. L'accès à la mémoire d'un autre processeur (dit voisin) s'effectue à travers un échange de messages sur le réseau entre les deux processeurs.

   Les algorithmes utilisés dans ce cas doivent minimiser ces communications en évitant les échanges inutiles afin d'optimiser le temps de traitement.

   Ce type de systèmes présente un ensemble d'avantages, notamment :

   - Chaque processeur est indépendant et a un accès rapide à son espace mémoire.
   - La mémoire est évolutive en fonction du nombre de processeurs.
   - Son coût est réduit et facile à construire.

   Ses inconvénients sont moins nombreux :

   - La gestion des communications doit être effectuée par le programmeur, ce qui génère une complexité des algorithmes, un risque d'erreurs et une source de perte de performances.
   - Puisque les communications passent via le réseau, les performances seront parfois au prix d'un réseau d'interconnexion coûteux.
2. Pour les systèmes à mémoire partagée : Chaque processeur a sa propre mémoire locale dans laquelle une partie de la mémoire globale sera copiée. Cette dernière est uniforme et visible par tous les processeurs. La gestion de l'ensemble de ces mémoires locales est effectuée soit par le matériel, le logiciel ou manuellement par l'utilisateur.

   Ce système présente deux grands avantages : la simplicité de sa programmation et la rapidité du partage des données entre les tâches.

   Par contre c'est le programmeur qui est responsable de la validité des synchronisations puisque le dit système n'est pas évolutif et donc ne favorise pas le

passage à l'échelle des accès en mémoire (tous les processeurs ne peuvent pas accéder à la mémoire globale en même temps).

3. Quant aux systèmes à mémoire hybride partagée distribuée : Ce sont des systèmes conçus pour bénéficier des avantages des deux précédents systèmes : la simplicité des architectures partagées et le coût réduit des architectures distribuées. Ceci peut être effectué par exemple en utilisant des systèmes à mémoire virtuellement partagée ou des langages à espace global adressable.

1.2.2 Efficacité du parallélisme

La principale raison pour laquelle nous utilisons le parallélisme est d'accélérer les calculs et de gagner du temps i.e. le temps mis pour effectuer un calcul. Souvent en informatique, pour un problème donné, il n'existe pas une seule solution, mais une multitude de façons pour implémenter la solution.

Pour évaluer le gain lié à la parallélisation d'un programme, on utilise le facteur d'accélération (*speedup*) et l'efficacité.

Le facteur d'accélération S(N) est défini comme le ratio du temps d'exécution du programme sur un processeur au temps d'exécution du même programme sur N processeurs (voir formule 1-1).

$$S(N) = \frac{t(1)}{t(N)} \qquad (1\text{-}1)$$

Où t(N) est le temps d'exécution sur N processeurs.

L'efficacité E(N) correspond alors au facteur d'accélération rapporté à un processeur (voir formule 1-2).

.

$$E(N) = \frac{S(N)}{N} \qquad (1\text{-}2)$$

1.2.3 Lois du parallélisme

Depuis les années soixante, plusieurs chercheurs se sont intéressés à la notion du parallélisme et ont proposé des lois, dont les plus connues sont :

1. La loi d'Amdahl (1967) : énoncée par GENE Amdahl dans [9] basée sur l'idée qu'une application est composée d'une partie intrinsèquement séquentielle et une partie décomposable en parties indépendantes (soit parallèlisable); la petite partie de l'application qui ne peut être parallélisée limite la vitesse globale du programme.

Soit P la fraction parallèlisable d'un programme et S sa fraction qui est inévitablement non-parallèlisable (soit séquentielle). On a alors le facteur d'accélération qui devient (voir formule 1-3).

$$S(N) = \frac{P+S}{\frac{P}{N}+S} = \frac{1}{P/N+S} \qquad (1\text{-}4)$$

2. La loi de Gustafson (1988) : proposée par ses deux inventeurs John L.Gustafson et Edwin H. Barsis dans [10], elle est basée sur la taille du problème (qui n'est pas fixe). Plus la taille d'un problème (quantité des données) est grande, plus le parallélisme est intéressant, puisqu'en augmentant le nombre d'unités de calcul ceci permet de traiter plus de données en un temps équivalent. Supposant que la taille d'un problème augmente au fur et à mesure avec le nombre d'unités de calculs (processeurs). La partie parallélisable est juste celle concernant les données.
   Avec le temps séquentiel= S + a * N
   Et le temps parallèle = S + a * N / N = S + a

Le facteur d'accélération devient (voir formule 1-4) :

$$S(N) = \frac{S + a * N}{S + a} \qquad (1\text{-}5)$$

Dans ce cas, le facteur d'accélération qui tend vers l'infini quand N tend vers l'infini.

3. La métrique de Karp-Flatt (1990) : Portant le nom de ses deux inventeurs Alan H. Karp et Horace P. Flatt, est un instrument de mesure de la parallélisation d'un système parallèle [11]. Elle prend en considération les deux précédentes lois et se présente plus efficace avec d'autres avantages, notamment :
    - La prise en compte des coûts de la parallélisation ainsi que la détection d'autres fuites et inefficacités non prises en compte par les autres lois.
    - Soit l'accélération AC obtenue pour N>1 processeurs, le calcul de la métrique de Karp–Flatt E(N) se fait comme suit (voir la formule 1-5) :

$$E(N) = \frac{\frac{1}{AC} - \frac{1}{N}}{1 - \frac{1}{N}} \qquad (1\text{-}6)$$

Plus la valeur de la métrique est petite, meilleure est la parallélisation.

1.2.4 Niveaux du parallélisme

Dans le parallélisme, chaque technique utilisée est définie par la taille du plus petit élément de l'ensemble des composantes à paralléliser, dont nous citons quatre types :

1. Parallélisme au niveau des instructions (en anglais "Instruction-level parallelism")
    - Représente plusieurs instructions du même flux d'instructions pouvant être exécutées simultanément.
    - Généré et géré par du matériel ou par le compilateur.
    - Limité en pratique par les dépendances de données et de contrôle.
2. Parallélisme au niveau des tâches (en anglais "Task-level parallelism")
    - Représente plusieurs threads ou séquences d'instructions de la même application pouvant être exécutés simultanément.

- Généré par le compilateur ou l'utilisateur et géré par le compilateur et le matériel.
- Limité dans la pratique par les frais généraux de communication ou de synchronisation et par les caractéristiques de l'algorithme.

3. Parallélisme au niveau des données (en anglais "Data-level parallelism")
    - Représente les instructions d'un seul flux fonctionnant simultanément sur plusieurs données.
    - Limité par des schémas de manipulation de données non réguliers et par la bande passante mémoire.
4. Parallélisme au niveau des transactions (en anglais "Transaction-level parallelism")
    - Plusieurs threads ou processus de différentes transactions peuvent être exécutés simultanément.
    - Limité par les frais généraux liés à la concurrence.

1.2.5 Stratégies du parallélisme

Le calcul parallèle consiste en l'utilisation simultanée de plusieurs ressources informatiques pour résoudre un problème de calcul en le décomposant en parties distinctes. Ce calcul peut se produire sur une seule machine ainsi que sur plusieurs machines. Le traitement sur une seule machine inclut les ordinateurs utilisant des processeurs multi-cœurs, multiprocesseurs et GPU avec plusieurs éléments de traitement.

Il existe une multitude d'exemples de machines incluant des grappes (clusters), des grilles (grids) et des nuages (clouds). En outre, les stratégies de parallélisation peuvent être classées sur la base de leur plate-forme de mise en œuvre [12], comme décrit ci-dessous:

1.2.5.1 Stratégies du parallélisme basées sur le CPU

Ces stratégies ont l'avantage d'accéder à plusieurs cœurs physiques ou virtuels avec un ou plusieurs processeurs. Ces approches comprennent:

- Hadoop MapReduce

Le modèle de programmation MapReduce a été établi par Google pour le traitement de données de grande taille, ce qui implique une parallélisation sur chaque CPU (ou sur un seul)

avec la distribution de différentes données. L'implémentation exécute les opérations de distribution de données, de parallélisation, d'équilibrage de charge et de tolérance aux pannes en tant que processus interne, tandis que l'utilisateur doit effectuer uniquement des opérations simples.

Le mappeur nécessite une paire d'entrée et produit un intermédiaire paire « clé / valeur ». Ensuite, toutes les valeurs intermédiaires contenant la même clé intermédiaire sont regroupées par la bibliothèque MapReduce et envoyées au réducteur. Le réducteur combine ensuite les valeurs correspondantes associées à la clé intermédiaire pour fusionner l'ensemble des valeurs.

- Boîte à outils de calcul parallèle MATLAB

MATLAB a fourni un mode de parallélisation simple en produisant une boîte à outils pour l'informatique parallèle. Elle est facile à utiliser, mais entraîne un coût élevé lors de l'achat.

Dans le pool parallèles, les variables sont accessibles à n'importe quel agent. La tâche de base reste donc d'initialiser le pool d'outils parallèle et de mentionner «la boucle pour» qui doit être parallélisée.

- Paquet parallèle R

R est un excellent langage open source doté de compétences statistiques et graphiques. Les développeurs ont conçu plusieurs progiciels de calcul parallèle dans R, parmi lesquels «foreach» et «doParallel» sont largement utilisés.

De plus, les codes C ++ sont intégrés dans R pour exécuter le calcul parallèle résultant du package Rcpp.

- Julia: «Parallel for» et MapReduce

Julia est un langage de programmation open-source moderne, fonctionnel et expressif, doté de remarquables capacités d'abstraction et de méta-programmation. Julia a été conçue dans le but de contribuer à une puissante parallélisation. Si chaque itération parallèle nécessite peu d'évaluations, alors «@parallel for» de Julia demeure le plus approprié.

Il est essentiellement créé pour l'attribution de petites tâches à chaque travailleur. Mais pour plusieurs variables de contrôle ou pour les modèles avec plusieurs choix discrets, «@parallel for» réduit la vitesse. Néanmoins, la fonction MapReduce dans Julia peut se révéler être une

excellente approche. Elle accepte les entrées sous forme de fonction et de vecteur de valeurs pour évaluer cette fonction.

- Module de calcul parallèle en Python

Python est un langage polyvalent, open source, interprété et flexible, contenant plusieurs modules. Dont, la fonction parallèle, une fonction du module Joblib semblable à une carte, est très populaire. Dans le pool parallèle, toutes les variables déclarées sont globalement observables et modifiables par les manipulateurs du langage.

- OpenMP avec C ++

C ++ est un langage compilé avec une vitesse remarquablement excellente, enrichi de robustesse et de flexibilité.

OpenMP est l'un des outils les plus simples en C ++ pour effectuer un calcul parallèle sur des systèmes de mémoire partagée multi-cœurs ou multiprocesseurs.

- MPI

Les processus parallèles exécutés sur des systèmes distribués, notamment des processeurs multi-cœurs établissent une communication via MPI. MPI est une bibliothèque avec un ensemble d'appels de fonction et de codes portables pour prendre en charge l'optimisation des performances.

1.2.5.2 Stratégies du parallélisme basées sur GPU

Les dix dernières années ont vu la popularité croissante de la parallélisation basée sur GPU. Le processeur graphique a des milliers de cœurs installés et la puissance de plusieurs processeurs dans un processeur.

Toute stratégie de parallélisation basée sur le CPU peut également être implémentée dans le processeur graphique. Les schémas de parallélisation les plus populaires basés sur GPU sont:

- CUDA

CUDA est un modèle de calcul parallèle qui permet la poursuite de calcul parallèle sur le processeur graphique de l'ordinateur en C, C ++, Fortran et Python.

En novembre 2006, nVIDIA™ a présenté l'architecture de calcul parallèle polyvalente appelée CUDA™, qui convient également au calcul massivement parallèle. Pour développer

les programmes parallèles, un GPU compatible et le CUDA™ SDK suffisent. L'utilisateur doit d'abord définir les fonctions devant être exécutées sur le GPU, puis l'allocation de mémoire des variables. Dès lors, le processus commence à partir de l'initialisation.

- OpenACC

OpenACC a une architecture analogue à celle d'OpenMP, bien qu'il soit encore au stade de développement. Il est construit en ajoutant simplement un code aux codes de série. Le code est portable pour GPU, CPU ou leur hybridation, peut donc fonctionner comme OpenMP, MPI ainsi que le calcul basé sur GPU.

En plus de ces approches, quelques approches constituant des options intermédiaires au-delà du CPU et du GPU, c'est-à-dire les processeurs d'hôte amorçables Intel Xeon Phi; TPU; et FPGA. Les approches MPI et OpenACC peuvent facilement adapter la mise en oeuvre d'Intel Xeon Phi. TPU est à un stade précoce de développement et les FPGA ne sont pas rentables. Par conséquent, ces trois approches se révèlent être peu populaires.

1.2.6 Modèles du parallélisme

Les processeurs inclus dans le calcul parallèle ne peuvent pas effectuer simultanément une interaction et échanger des informations. Ils communiquent via des stratégies spécifiques connues sous le nom de «modèles parallèles», basées sur des topologies de réseau, comme indiqué dans la Figure 1-1 [12]. Ces modèles parallèles sont classés en quatre stratégies de communication principales : (i) maître-esclave; (ii) à grain grossier (modèle d'île); (iii) grain fin (modèle cellulaire) et (iv) hybride.

Le modèle maître-esclave fonctionne comme la topologie en étoile. Les évaluations de fitness fonctionnent en parallèle sur les processeurs esclaves, et le processeur maître contrôle les opérations et les générations de tous les n processeurs esclaves, comme indiqué sur la Figure 1-1 (a). Les modèles à grain grossier et à grain fin explorent certaines restrictions de voisinage pour les communications entre les processeurs.

Les modèles à grain fin sont également appelés modèles cellulaires, les modèles maître-esclave sont également connus sous le nom de topologie en étoile et les modèles à grain

grossier sont également connus en tant que modèles d'île ou topologie en anneau. Cela rend l'algorithme correspondant plus robuste et efficace.

Dans le modèle à grains grossiers, la population entière est divisée en n sous-populations mutuellement indépendantes qui représentent une unité de traitement, appelée «île», comme indiqué sur la Figure 1-1 (b). Certains individus sont échangés entre les îles selon une stratégie de migration (l'échange d'individus est appelé migration).

Ce modèle de communication est contrôlé par plusieurs paramètres tels que: la stratégie de migration, la population de migration et la taille des sous-populations. Dans un modèle à grain fin, les individus sont disposés dans une grille 2D dans laquelle chaque individu a 4 voisins, comme le montre la Figure 1-1 (c). La communication entre ces voisinages peut se produire de plusieurs manières ; par conséquent, l'échange d'informations est retardé entre les processeurs non voisins. Les modèles hybrides sont l'hybridation de deux modèles discutés ou plus.

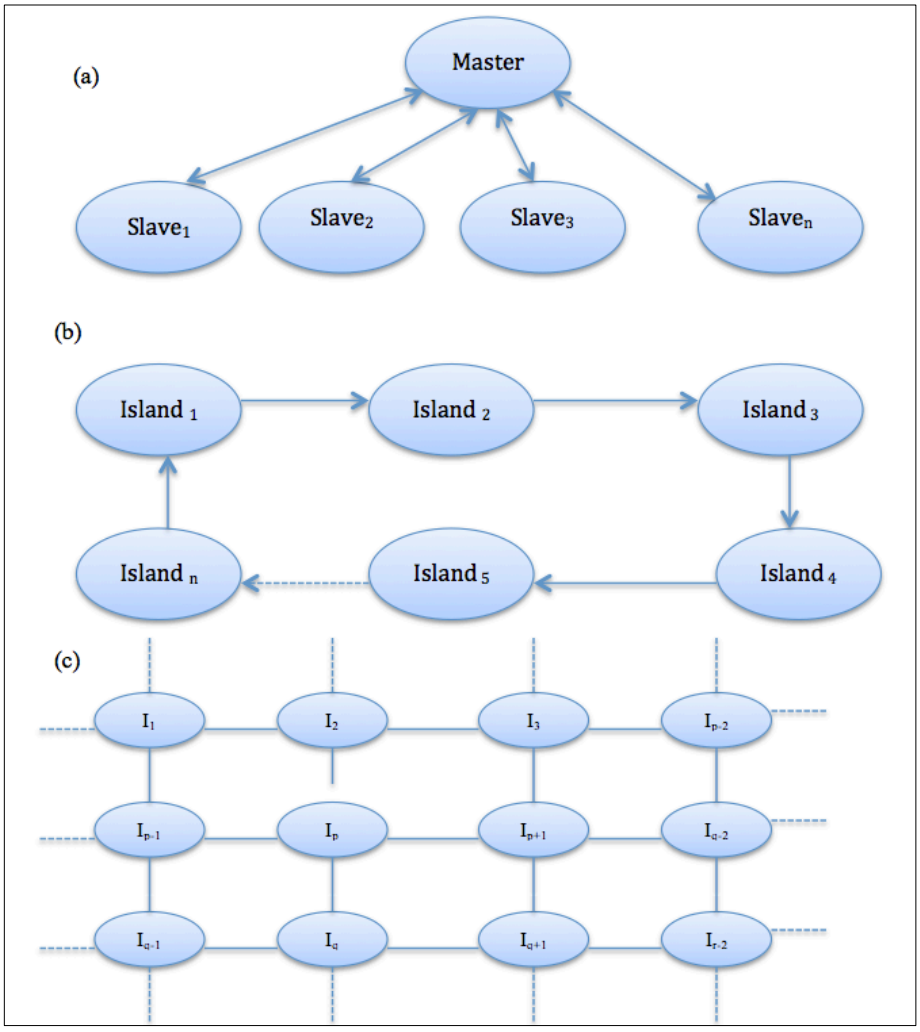


Figure 1-1 Modèles parallèles (a) Maître-esclave, (b) à grain grossier, (c) à grain fin.

1.2.7 Programmation concurrente et synchronisation

En programmation, la concurrence consiste à traiter plusieurs tâches en même temps. Elle englobe un ensemble de concepts, entre autres : la coopération, la communication et l'échange d'informations.

Généralement, dans la programmation concurrente (avec les processus ou les threads), il existe deux façons pour communiquer : à travers un échange de messages et de signaux pour les processus ou via des variables partagées pour les threads.

1.2.7.1 Processus lourd vs. processus léger

Les processus comme les threads représentent l'exécution d'un ensemble d'instructions du langage machine d'un processeur et qui permettent d'effectuer des tâches en parallèles. Toutefois, il existe un nombre de distinctions entre les processus et les threads.

- **Processus lourd (Process en anglais)**
  - Un processus est une entité dynamique représentant un programme en cours d'exécution sur un processeur.
  - Créé suite à 4 situations: au démarrage du système, suite à un appel système de création par un processus, une demande de création effectuée par un utilisateur, ou un "Batch job" (soumission d'un travail à un cluster…).
  - Comme leur création, la terminaison d'un processus s'effectue suite à 4 situations: Une fin normale, une fin anormale, une erreur ou tué par un autre processus (kill).
  - Chaque processus possède sa propre mémoire virtuelle privée (indépendante des autres processus).
  - Deux processus souhaitant collaborer doivent passer via un mécanisme d'échange de messages (par exemple: les pipes sous Unix), puisqu'ils n'ont pas un espace de mémoire commun.
  - Le contexte d'un processus est lourd: il représente un ensemble de code, registres, pile, fichiers, variables, compteur d'insctruction, vecteur d'interruptions, etc. Créer et activer un processus ainsi qu'effectuer un changement de contexte se révèle donc être très coûteux !
  - Les processus peuvent résider dans plusieurs états, entre autres: Actif ou en exécution, ce qui indique qu'il est utilisé par le CPU, éligible ou prêt représente qu'il est en attente du CPU, bloqué indique que ce processus est en attente d'un évènement extérieur (ex: interruption) ou volontairement arrêté. Voir aussi Figure 1-2.

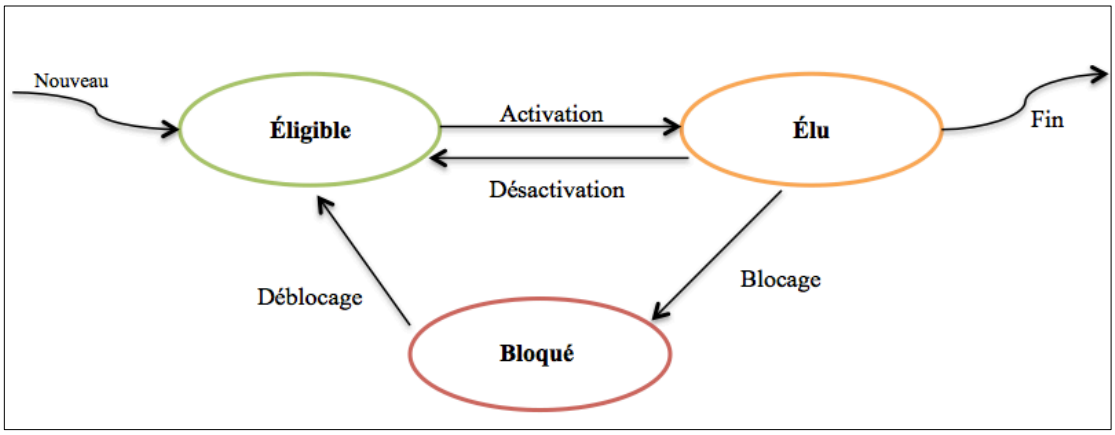


Figure 1-2 Diagramme d'état d'un processus.

Un processus peut jouer le rôle d'un système d'exploitation en lançant des sous tâches internes au processeur (nommées « flux d'exécution »).

- **Processus léger ou Fil d'exécution, tâche (Thread en anglais)**
  - Un thread est une partie d'un processus, généralement une fonction en cours d'exécution gérable indépendamment par l'ordonnanceur (Scheduler) du système d'exploitation.
  - Un processus peut contenir un ou plusieurs threads, on parle respectivement du single threading et de multithreading. Ces threads partagent un ensemble de ressources principalement la mémoire - voir aussi Figure 1-3.
  - La communication entre les threads se fait à travers les variables partagées (du fait que les threads partagent le même espace mémoire).
  - Comme son nom l'indique, « processus léger », le contexte d'un thread s'avère être moins coûteux que celui d'un processus.

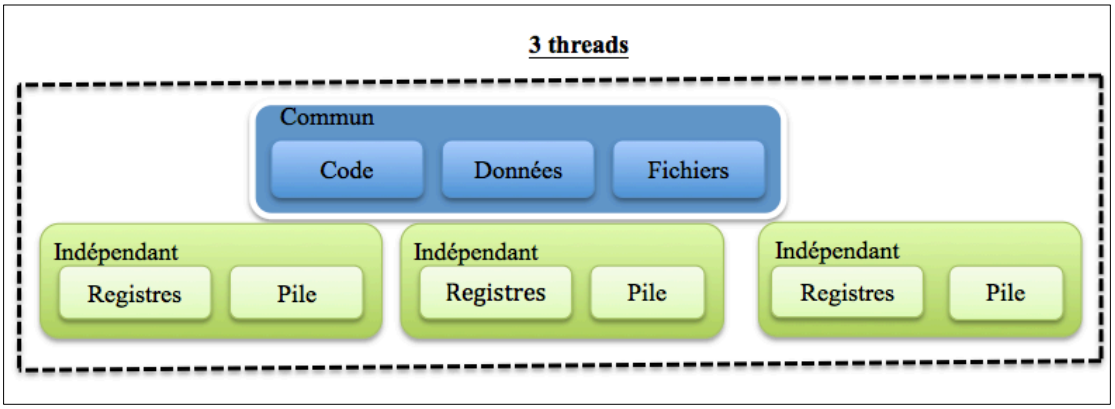


Figure 1-3 Représentation de trois threads (1 processus = n threads).

1.2.7.2 Synchronisation

La concurrence s'impose dans tous les programmes réalisés en interaction avec notre monde réel. Le problème se pose au moment de l'accès à une ressource partagée, ce qui provoque des incohérences. Pour remédier à ceci, l'utilisation d'un ensemble de mécanismes assurant l'exécution correcte de l'ensemble des processus coopérants s'impose.

La synchronisation est donc un mécanisme utilisé pour la gestion des contraintes sur l'ordonnancement des tâches dans le temps. Elle permet à des processus non disjoints de s'exécuter concurremment et de produire de bons résultats.

Deux types de synchronisation est à distinguer:

1. Synchronisation des processus ou des tâches : ce type de synchronisation est le moins utilisé, il représente un mécanisme visant à bloquer l'exécution des différents processus à des points précis de leur programme de manière à ce que tous les processus passent les étapes bloquantes au moment prévu par le programmeur.
2. La synchronisation de données : elle représente le type de synchronisation le plus connu, son rôle est de conserver la cohérence entre différentes données dans un environnement multitâche. Les problèmes liés à la synchronisation rendent toujours la programmation concurrente plus difficile.

1.3 Optimisation mathématique

L'optimisation est un outil important pour la prise de décision et l'analyse des systèmes physiques. En terme mathématique, un problème d'optimisation désigne la problématique de trouver la meilleure solution parmi l'ensemble des solutions possibles, en maximisant ou minimisant une fonction par rapport à un ensemble. La fonction objectif permet de comparer les différents choix pour déterminer lequel serait le mieux adapté.

Plus formellement, nous définissons le problème d'optimisation ainsi: $\underset{x \in S}{optimiser}\, f(x)$.

Où *optimiser* signifie min ou max $f : R^n \rightarrow R$ désigne la fonction objectif, et S $\subseteq$ R est l'ensemble réalisable, à savoir l'ensemble de tous les choix admissibles pour x.

Dans ce qui suit, nous ferons référence aux problèmes de minimisation. En effet, la solution optimale d'un problème de maximisation : $\max_{x \in S} f(x)$

Coïncider avec les solutions optimales du problème de minimisation : $\min_{x \in S} -f(x)$

Et nous avons : $\max_{x \in S} f(x) = \min_{x \in S}(-f(x))$

L'ensemble réalisable S est un sous-ensemble de R^n et donc $x = (x_1, x_2, ..., x_n)^T$

Avec T qui représente le vecteur de variables de dimension n et f est une fonction de n valeurs réelles $f = (x_1, x_2, ..., x_n)$.

1.3.1 Étapes du processus de l'optimisation

Le processus d'optimisation passe par un ensemble d'étapes, qui permettent la bonne résolution de la problématique à optimiser [13].

1.3.1.1 Construire un modèle

La première étape du processus d'optimisation consiste à construire un modèle approprié. La modélisation est le processus d'identification et d'expression mathématique de l'objectif, des variables et des contraintes du problème.

Un objectif est une mesure quantitative de la performance du système que nous voulons minimiser ou maximiser. Par exemple, dans le secteur de la fabrication, nous pouvons maximiser les profits ou minimiser les coûts de production, alors que pour ajuster les données expérimentales à un modèle, nous voudrons minimiser l'écart total entre les données observées et les données prédites.

Les variables sont les composantes du système pour lesquelles nous voulons trouver des valeurs. En fabrication, les variables peuvent être la quantité de chaque ressource consommée ou le temps consacré à chaque activité, alors que dans l'ajustement des données, les variables seraient les paramètres du modèle.

Les contraintes quant à elles sont les fonctions qui décrivent les relations entre les variables et définissent les valeurs autorisées pour ces dernières. En fabrication, la quantité de ressources consommées ne peut dépasser la quantité disponible.

1.3.1.2 Déterminer le type de problème

La deuxième étape du processus d'optimisation consiste à déterminer à quelle catégorie d'optimisation appartient le problème à optimiser. Cette étape est très importante car elle permet de classer le problème d'optimisation afin de faciliter le choix de la solution appropriée. (Le lecteur peut se référer à [13] pour d'amples informations sur les indications qui permettent d'identifier et de classer chaque problème d'optimisation).

1.3.1.3 Sélectionner la méthode d'optimisation

La troisième étape du processus d'optimisation consiste à sélectionner un algorithme approprié au type de problème d'optimisation à résoudre. C'est une étape perçue d'une grande importance dans le processus d'optimisation, du fait que la bonne résolution de n'importe quelle problématique est due en grande partie au choix adéquat de l'outil à utiliser (approprié à la problématique).

Dans le domaine de l'optimisation, il existe une grande variété d'outils, utilisés dans différents secteurs, selon le problème à optimiser.

Quant à nous, notre intérêt s'est focalisé sur les méthodes d'optimisation approchées, particulièrement les méta-heuristiques pour la résolution des problèmes d'optimisation complexes.

1.3.2 Classification des types d'optimisation

Comme indiqué précédemment, la classification des problèmes d'optimisation est une étape importante du processus d'optimisation, car les algorithmes de résolution des problèmes d'optimisation sont adaptés à un type de problème particulier. Il existe plusieurs classifications des problèmes d'optimisation [13], dont nous citons ci-dessous quelques unes :

1.3.2.1 Optimisation continue et optimisation discrète

Certains modèles n'ont de sens que si les variables prennent des valeurs d'un ensemble discret, souvent un sous-ensemble d'entiers ou de binaires, alors que d'autres modèles contiennent des variables pouvant prendre n'importe quelle valeur réelle. Les modèles avec des variables discrètes sont des problèmes d'optimisation discrète; tandis que, les modèles à variables continues sont des problèmes d'optimisation continue.

Les problèmes d'optimisation continue ont tendance à être plus faciles à résoudre que les problèmes d'optimisation discrète; la régularité des fonctions signifie que les valeurs de la fonction objectif et de la fonction de contrainte en un point x peuvent être utilisées pour déduire des informations sur les points situés dans un voisinage de x. Cependant, les améliorations apportées aux algorithmes, associées aux progrès de la technologie informatique, ont considérablement accru la taille et la complexité des problèmes d'optimisation discrète pouvant être résolus efficacement. Les algorithmes d'optimisation continue sont importants dans l'optimisation discrète car de nombreux algorithmes d'optimisation discrète génèrent une séquence de sous-problèmes continus.

1.3.2.2 Optimisation sans contrainte et optimisation avec contrainte

Une autre distinction importante entre les problèmes dans lesquels il n'y a pas de contraintes sur les variables et les problèmes dans lesquels il y a des contraintes sur les variables. Les problèmes d'optimisation sans contrainte se posent directement dans de nombreuses

applications pratiques; ils se posent également lors de la reformulation de problèmes d'optimisation sous contraintes dans lesquels les contraintes sont remplacées par un terme de pénalité dans la fonction objectif. Les problèmes d'optimisation sous contrainte proviennent d'applications dans lesquelles les variables sont explicitement contraintes. Les contraintes imposées aux variables peuvent varier considérablement, allant de simples bornes à des systèmes d'égalités et d'inégalités modélisant des relations complexes entre les variables. Les problèmes d'optimisation sous contrainte sont compliqués. Ils nécessitent l'utilisation d'algorithmes dédiés et peuvent être classés selon la nature des contraintes (par exemple, linéaires, non linéaires, convexes) et la régularité des fonctions (par exemple, différenciables ou non différenciables).

1.3.2.3 Optimisation mono-objectif et optimisation multi-objectifs

La plupart des problèmes d'optimisation ont une fonction objectif unique. Cependant, il existe des cas intéressants lorsque les problèmes d'optimisation n'ont pas de fonction objectif ou comportent plusieurs fonctions objectif. Les problèmes de faisabilité sont des problèmes dans lesquels l'objectif est de trouver des valeurs pour les variables qui répondent aux contraintes d'un modèle sans objectif particulier à optimiser. Les problèmes de complémentarité sont omniprésents en ingénierie et en économie. Le but est de trouver une solution qui réponde aux conditions de complémentarité. Des problèmes d'optimisation à objectifs multiples se posent dans de nombreux domaines, tels que l'ingénierie, l'économie et la logistique, lorsque des décisions optimales doivent être prises en présence de compromis entre deux objectifs contradictoires ou plus. Par exemple, développer un nouveau composant peut impliquer de minimiser le poids tout en maximisant la solidité; choisir un portefeuille peut impliquer de maximiser le rendement attendu tout en minimisant les risques. En pratique, les problèmes à objectifs multiples sont souvent reformulés en problèmes à objectif unique soit en formant une combinaison pondérée des différents objectifs, soit en remplaçant certains objectifs par des contraintes.

1.3.2.4 Optimisation déterministe et optimisation stochastique

En optimisation déterministe, il est supposé que les données pour le problème à optimiser sont connues avec précision. Cependant, pour de nombreux problèmes réels, les données ne peuvent pas être connues avec précision pour diverses raisons. La première raison est due à une simple erreur de mesure. La deuxième raison, plus fondamentale, est que certaines données représentent des informations sur le futur (par exemple, la demande d'un produit ou un prix pour une période future) et ne peuvent tout simplement pas être connues avec certitude. En optimisation sous incertitude, ou optimisation stochastique, l'incertitude est incorporée dans le modèle. Des techniques d'optimisation robustes peuvent être utilisées lorsque les paramètres ne sont connus que dans certaines limites. Le but est de trouver une solution réalisable pour toutes les données et optimale dans un sens. Les modèles de programmation stochastiques tirent profit du fait que les distributions de probabilité régissant les données sont connues ou peuvent être estimées; l'objectif est de trouver une stratégie qui soit réalisable pour toutes les instances de données possibles (ou presque) et d'optimiser les performances attendues du modèle.

1.3.3 Méthodes méta-heuristiques

Les méta-heuristiques sont une famille d'algorithmes stochastiques, apparues à partir de l'année 1980, destinées à la résolution des problèmes d'optimisation difficiles.

Elles permettent d'obtenir une valeur approchée de la solution optimale en un temps raisonnable. Leur particularité réside dans leur fléxibilité, càd que celles-ci sont adaptables à la résolution d'un ensemble de problèmes dans différents domaines sans avoir à modifier le principe de base de la méthode (d'où le qualificatif 'méta').

Les méta-heuristiques sont souvent inspirées de processus naturels qui relèvent de la physique (l'algorithme du recuit simulé), de la biologie de l'évolution (les algorithmes génétiques) ou encore de l'éthologie (les algorithmes de colonies de fourmis ou l'optimisation par essaim particulaire).

L'ensemble des méta-heuristiques proposées dans la littérature sont partagées en deux classes: des métaheuristiques à base de solution unique et des méta-heuristiques à base de

population de solutions [14].

Les méta-heuristiques à base de solution courante unique manipulent un point, et décident à chaque itération quel sera le point suivant. Elles débutent la recherche avec une seule solution initiale et se basent sur la notion du voisinage pour améliorer la qualité de la solution courante.

De nombreuses méthodes à base de solution unique ont été proposées dans la littérature. Parmi lesquelles: le recuit simulé, la recherche tabou, la recherche à voisinage variable, la recherche locale réitérée, la recherche locale guidée, la descente …etc.

Les méthodes à base de population de solutions: qui manipulent une population de points, et à chaque itération un nouveau jeu de points est choisi. Elles débutent la recherche avec une multitude de solutions. Elles s'appliquent sur un ensemble de solutions afin d'en extraire la plus adéquate (l'optimum global) qui représentera la solution du problème traité.

L'utilisation d'un ensemble de solutions au lieu d'une seule solution permet une bonne diversité de la recherche et augmente la possibilité d'émergence de solutions de meilleure qualité.

Une grande variété de méthodes basées sur une population de solutions a été proposée dans la littérature, commençant par les algorithmes évolutionnaires, passant par les algorithmes génétiques et arrivant aux algorithmes à base d'intelligence par essaims (l'algorithme d'optimisation par essaim de particules, l'algorithme de colonies de fourmis, l'algorithme de colonies d'abeilles, …etc.

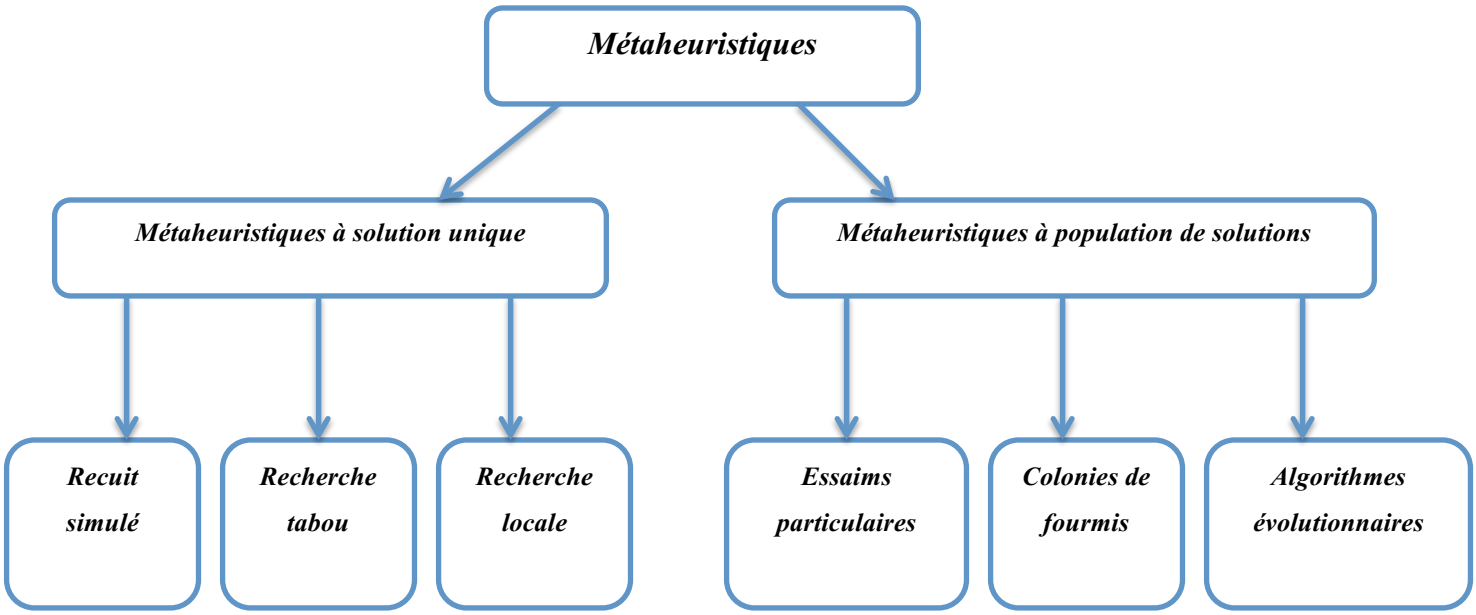


Figure 1-4 Exemples de méta-heuristiques à solution unique et à population de solutions.

1.3.3.1 Méthode d'essaim particulaire (PSO)

L'optimisation par essaim particulaire (en anglais : Particle Swarm Optimization) est une méta-heuristique d'optimisation stochastique à base de population de solutions, proposée en 1995 par James Kennedy (socio-psychologue) et Russel Eberhart (ingénieur électricien) pour la résolution des problèmes d'optimisation, plus particulièrement les problèmes à variable continue [15].

Elle est basée sur le comportement social des individus évoluant en essaim, càd, les « interactions sociales » entre des « agents » appelés « particules » représentant un « essaim », dans le but d'atteindre un objectif donné dans un espace de recherche commun où chaque particule a une certaine capacité de mémorisation et de traitement de l'information.

Dans PSO le comportement social est modélisé par une équation mathématique permettant de guider les particules durant leur processus de déplacement [15].

Le déplacement d'une particule est influencé par trois composantes: la composante d'inertie, la composante cognitive et la composante sociale. Chacune de ces composantes reflète une partie de l'équation, Figure 1–5 [1]:

1. La composante d'inertie: la particule tend à suivre sa direction courante de déplacement;
2. La composante cognitive: la particule tend à se diriger vers le meilleur site par lequel elle est déjà passée;
3. La composante sociale: la particule tend à se diriger vers le meilleur site atteint par ses voisines.

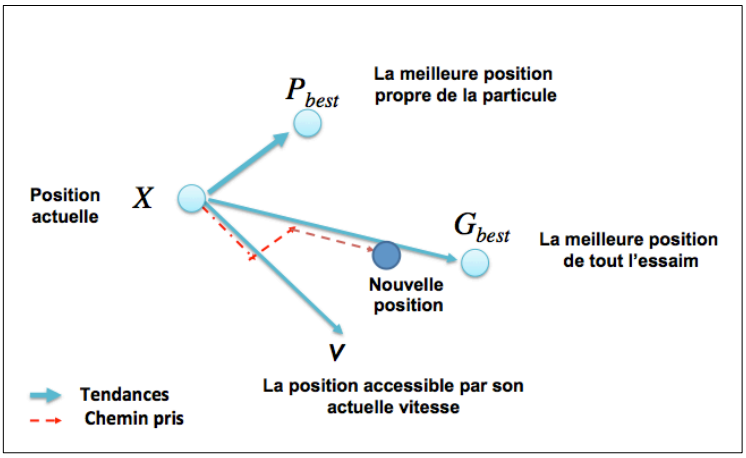


Figure 1-5 Déplacement de la particule.

Formalisation et algorithme

Une particule i de l'essaim dans un espace de recherche de dimension D est caractérisée, à l'instant t, par:

- X: sa position dans l'espace de recherche,
- V: sa vitesse,
- Pb: la position de la meilleure solution par laquelle elle est passée,
- Pg: la position de la meilleure solution connue de tout l'essaim,
- f (Pb): la valeur de fitness de sa meilleure solution,
- f (Pg): la valeur de fitness de la meilleure solution connue de tout l'essaim,

Le déplacement de la particule i entre les itérations t et t+1 se fait selon les deux équations (1-6) et (1-7) [15] :

$$\begin{cases} V_{iD(t+1)} = V_{iD(t)} + C_1 r_1 (Pb_{iD(t)} - X_{iD(t)}) + C_2 r_2 (Pg_{iD(t)} - X_{iD(t)}) & (1\text{-}6) \\ X_{iD(t+1)} = X_{iD(t)} + V_{iD(t)} & (1\text{-}7) \end{cases}$$

Où :

- C_1 et C_2 : deux constantes qui représentent les coefficients d'accélération, elles peuvent être non constantes dans certains cas selon le problème d'optimisation posé [16] [17].
- r_1 et r_2 : deux nombres aléatoires tirés de l'intervalle [0,1].

L'algorithme de base de la méthode PSO proposé par [15], commence par une initialisation aléatoire des particules dans leur espace de recherche, en leurs attribuant une position et une vitesse initiales. À chaque itération de l'algorithme les particules se déplacent selon les équations (1) et (2) et les fonctions objectif (fitness) des particules sont calculées afin de pouvoir calculer la meilleure position de toutes Pg. La mise à jour des Pb et Pg est faite à chaque itération (voir Figure 1–6). Le processus est répété jusqu'à satisfaction du critère d'arrêt.

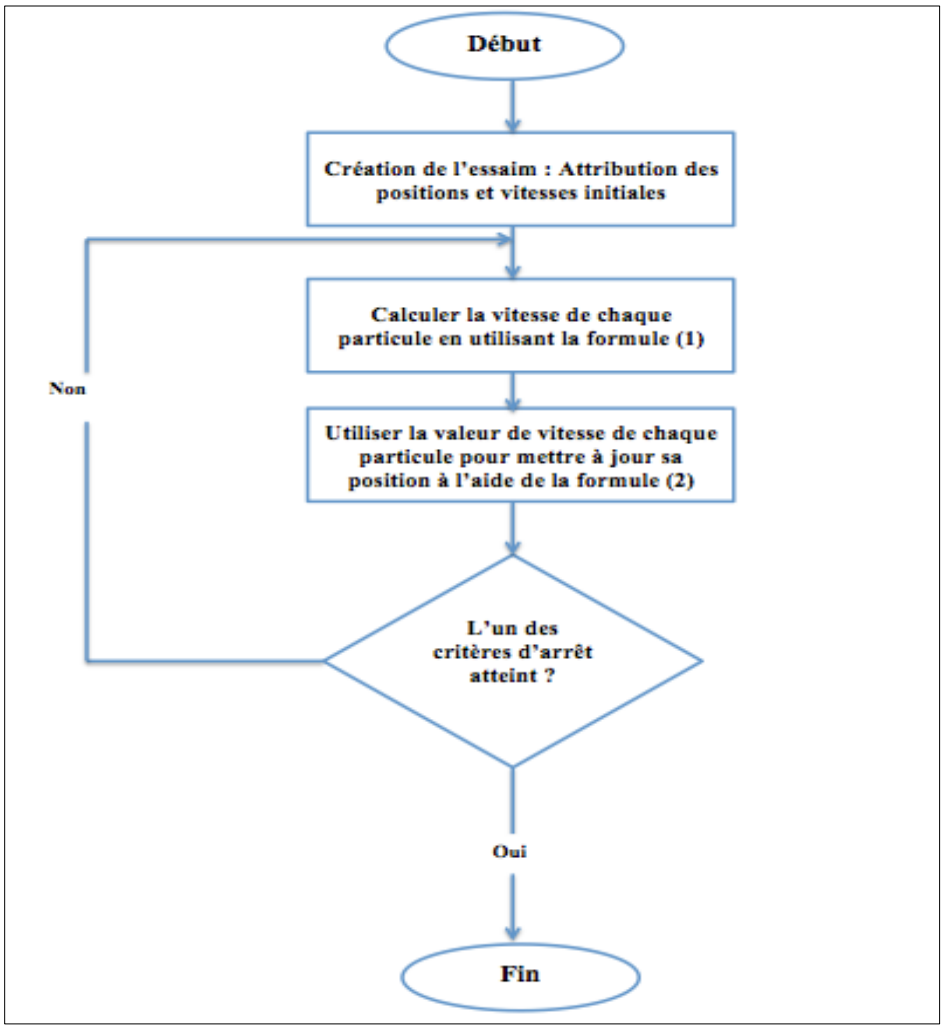


Figure 1-6 Organigramme de l'algorithme de base de la méthode PSO.

- **Paramètres PSO**

Comme toute méta-heuristique, PSO dispose d'un ensemble de paramètres qui interviennent et influencent sa performance. Le choix de ces paramètres reste critique et dépend généralement du problème posé [18] [19] mais a une grande influence sur la convergence de l'algorithme. Parmi les paramètres qui rentrent en ligne de compte nous pouvons citer :

- **Nombre des particules**

L'un des paramètres clés de la méthode PSO est le nombre de particules, ce dernier influence grandement la performance de l'algorithme, surtout au niveau du temps de calcul, puisque la

présence de chaque particule dans l'algorithme provoque un calcul : évaluation de la position et déplacement de la particule [20].

La quantité de particules allouées à la résolution d'un problème dépend de plusieurs paramètres, à savoir : la dimension du problème à optimiser (la taille de l'espace de recherche), le rapport entre les capacités de calcul de la machine et le temps maximum de recherche et surtout la complexité du problème d'optimisation.

Le choix d'une valeur adéquate pour ce paramètre n'est pas une tâche facile, puisqu'il n'y a pas de règle pour la déterminer, seule une expérimentation massive en faisant de nombreux essais permet de se doter de l'expérience nécessaire à l'appréhension de ce paramètre.

- **Dimension du problème**

Ce paramètre est d'une liaison directe avec le problème à optimiser, il représente l'espace de recherche et d'évolution des particules, et bien sur la dimension du problème impacte la convergence de l'algorithme, au niveau du temps de calcul : traiter un problème à 2 dimensions, ne prend pas le même temps qu'un problème à 30 dimensions, et aussi au niveau de la fiabilité des résultats : la précision des résultats à petite dimension n'est pas la même que celle liée aux problèmes à dimension élevée.

- **Disposition des particules**

Avant le démarrage de l'algorithme, les positions des particules ainsi que leurs vitesses initiales doivent être initialisés aléatoirement selon une loi uniforme sur [0..1]. Cette disposition initiale a une influence sur le déplacement prochain de chaque particule et donc la convergence de l'algorithme, surtout dans le cas de la constitution de voisinages géographiques. Cependant, il existe un ensemble de générateurs automatiques de positions, permettant d'attribuer des positions différentes à l'ensemble de l'essaim. Plusieurs études existent dans ce sens [21].

Le générateur de séquence de SOBOL est l'un des plus performants dans le domaine, pour une disposition homogène des particules dans un espace de dimension n suite à une étude faite par [22] qui ont utilisé une certaine séquence à faible écart pour initialiser les particules.

Les chercheurs ont utilisé les générateurs de séquences de Halton, Sobol et Faure pour initialiser l'essaim. Ils ont ensuite testé les variantes proposées à l'aide de six fonctions de test standard. Ils ont constaté que la performance de PSO avec l'initialisation Sobol est la meilleure parmi toutes les techniques.

- **Coefficients de confiance**

Ces deux paramètres sont utilisés dans la formule de déplacement des particules, ils sont variables et pondèrent les tendances de la particule à vouloir suivre son instinct de conservation ou son panurgisme.

Les variables aléatoires σ_1 et σ_2 peuvent être définies de la façon suivante :

$$\sigma_1 = C_1 r_1$$
$$\sigma_2 = C_2 r_2$$

Où r_1 et r_2 suivent une loi uniforme sur [0..1] et C_1 et C_2 : sont des constantes positives déterminées de façon empirique et suivant la relation $(C_1 + C_2) \leq 4$.

Vu l'utilisation importante de ces paramètres à chaque itération de l'algorithme PSO, plusieurs études ont été menées pour trouver leurs valeurs optimales [16] [17].

- **Critère d'arrêt**

Le critère d'arrêt représente l'une des clés de succès de l'algorithme PSO, choisir un critère d'arrêt optimal n'est pas une tâche facile, et ne se fait pas au hasard, mais doit être le résultat d'une étude approfondie de la problématique ainsi qu'une expérimentation massive. Ce paramètre diffère selon le problème d'optimisation posé et les contraintes définies par l'utilisateur, il est fortement conseillé de doter l'algorithme d'une porte de sortie puisque la convergence vers la solution optimale globale n'est pas garantie dans tous les cas de figure même si les expériences indiquent la grande performance de la méthode. De ce fait plusieurs études ont été menées dans ce sens [23], différentes propositions ont eu lieu: l'algorithme doit alors s'exécuter tant que l'un des critères de convergence n'a pas été atteint cela peut être: le nombre maximum d'itérations; l'optimum global est connu a priori, on peut définir une

"précision acceptable", la variation de la vitesse est proche de 0. D'autres critères d'arrêt peuvent être utilisés selon le problème d'optimisation posé et des contraintes définies par l'utilisateur.

- **Améliorations de la méthode PSO**

Depuis son apparition, l'algorithme PSO a eu un grand succès dans le domaine de l'optimisation, et a intéressé une grande communauté scientifique. Plusieurs améliorations ont été apportées à l'algorithme de base de la méthode, différentes variantes ont été proposées soit en l'hybridant avec d'autres méta-heuristiques, en le parallélisant, ou en ajoutant certains nouveaux paramètres [24] -[28].
Parmi les versions proposant l'ajout de certains paramètres :

- **Coefficient d'inertie**

Le coefficient d'inertie ω a été introduit par [29] pour contrôler l'influence de la direction de la particule sur le déplacement futur. Le but de l'introduction de ce paramètre est de réaliser un équilibre entre la recherche locale (exploitation) et la recherche globale (exploration). La formule (1-6) de calcul de la vitesse devient:

$$V_{iD(t+1)} = \omega\left[V_{iD(t)} + C_1 r_1 (Pb_{iD(t)} - X_{iD(t)}) + C_2 r_2 (Pg_{iD(t)} - X_{iD(t)})\right] \quad (1\text{-}8)$$

La valeur de ω est généralement constante, mais peut être variable dans certains cas, une grande valeur de ω est synonyme d'une grande amplitude de mouvement et donc d'exploration globale de l'espace de recherche. Les études menées par [29] indiquent une meilleure convergence pour ω entre 0.8 et 1.2. La détermination de la meilleure valeur de ce paramètre pour chaque algorithme se fait à travers des expérimentations numériques.
Dans [29], les auteurs ont proposé un coefficient d'inertie dynamique qui varie au cours du temps, et diminue linéairement au cours du processus de l'optimisation. Il commence par une valeur proche de 0.9 et descend linéairement pour arriver à 0.4.

- **Facteur de constriction**

Le facteur de constriction χ a été proposé par [30], dans le but d'améliorer la convergence de l'algorithme, de prévenir l'explosion de l'essaim et de contrôler la vitesse des particules afin d'échapper au problème de la divergence de l'essaim qui cause la convergence prématurée de l'algorithme. La formule de la vitesse (1-6) devient alors:

$$V_{iD(t+1)} = \chi\left[V_{iD(t)} + C_1 r_1 (Pb_{iD(t)} - X_{iD(t)}) + C_2 r_2 (Pg_{iD(t)} - X_{iD(t)})\right] \quad (1\text{-}9)$$

Où :

$$\chi = \frac{2}{\left| 2 - \sigma - \sqrt{\sigma^2 - 4\sigma} \right|}$$

Et :

$$\sigma = \sigma_1 + \sigma_2 \geq 4.0$$
$$\sigma_1 = C_1 r_1$$
$$\sigma_2 = C_2 r_2$$

Selon d'autres études menées par [31], dans certains cas, le coefficient de constriction seul ne permet pas la convergence vers la solution optimale pour un nombre d'itérations donné. Pour résoudre ce problème, il peut être intéressant de fixer Vmax = Xmax en plus du coefficient de constriction, ce qui permet d'améliorer les performances globales de l'algorithme.

- **Vitesse maximale**

C'est un paramètre qui a été proposé par [32], comme solution au problème de déviation des particules au cours de leur déplacement.

L'objectif été de limiter la vitesse des particules par l'intervalle [-Vmax, Vmax] dans le but de contrôler le déplacement de chaque particule dans l'espace de recherche.

L'introduction de Vmax a permis un meilleur contrôle de déplacement des particules pour une convergence plus optimale.

L'utilisation de ce paramètre a donné lieu à plusieurs publications [33] - [35].

- **Notion de voisinage**

Le voisinage constitue la structure du réseau social. Le voisinage d'une particule représente avec qui chacune des particules va pouvoir communiquer. Il existe deux principaux types de voisinages :

Le voisinage géographique: ce type de voisinage représente la proximité géographique, c'est la notion la plus naturelle du voisinage pour les essaims particulaires, les voisins sont considérés comme les particules les plus proches. Cependant, à chaque itération, les nouveaux voisins doivent être recalculés à partir d'une distance prédéfinie dans l'espace de recherche. C'est donc un voisinage dynamique qu'il convient de définir et d'actualiser à chaque itération.

Le voisinage social: ce type de voisinage représente la proximité sociale, les voisinages ne sont plus l'expression de la distance mais l'expression de l'échange d'informations, les voisins sont définis à l'initialisation et ne sont pas modifiés par la suite. Une fois le réseau des connexions sociales établi, il n y à pas besoin de le réactualiser. C'est donc un voisinage statique.

La modification de la formule de vitesse (1-6) est réalisée en utilisant un nouveau terme dans l'équation. Il a été introduit par [36], son illustration paraît dans Figure 1–7 [2] :

$$V_{iD(t+1)} = V_{iD(t)} + C_1 r_1 (Pb_{iD(t)} - X_{iD(t)}) + C_2 r_2 (Pg_{iD(t)} - X_{iD(t)}) + C_3 r_3 (Pn_{iD(t)} - X_{iD(t)}) \quad (1\text{-}10)$$

Où :

Pn: La meilleure position du voisinage.

C_3 : Le coefficient d'accélération, appelé aussi paramètre social.

r_3 : Nombre aléatoire tiré de l'intervalle [0,1].

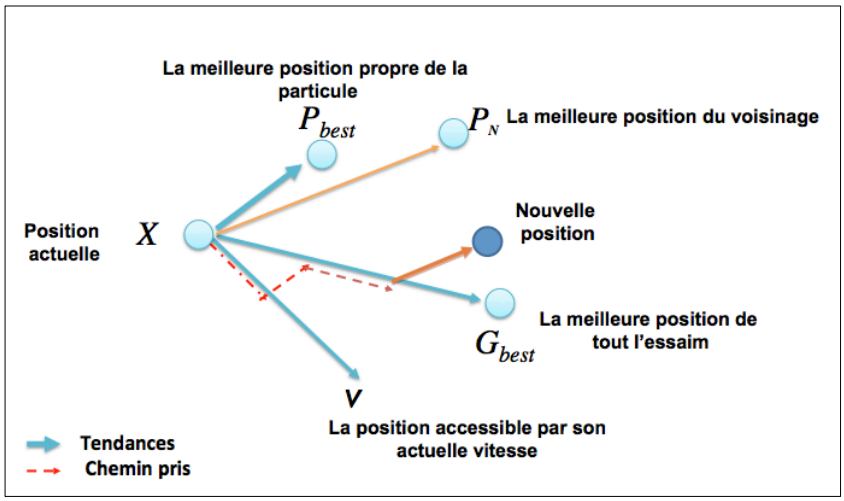


Figure 1-7 Déplacement de la particule avec la notion de voisinage.

- **Topologie de voisinage**

Le réseau de rapports entre toutes les particules est connu sous le nom de "la topologie de l'essaim". Le choix d'une topologie de voisinage à une importance cruciale, plusieurs études de topologies ont été menées à ce propos [37], différentes combinaisons ont été proposées dont les plus utilisées sont susmentionnées ci-dessous [2] :

Topologie en anneau (Figure 1–8 (a)): chaque particule est reliée à n particules, (généralement n = 3), chaque particule tend à se diriger vers la meilleure dans son voisinage locale.

Topologie en rayon (Figure 1–8 (b)): la communication entre les particules est faite via une particule centrale, seule cette dernière ajuste sa position vers la meilleure, s'il y à amélioration de sa position, l'information est alors propagée à ses congénères.

Topologie en étoile (Figure 1–8 (c)): chaque particule est reliée à toutes les autres, le réseau social est complet, càd. L'optimum du voisinage est l'optimum global.

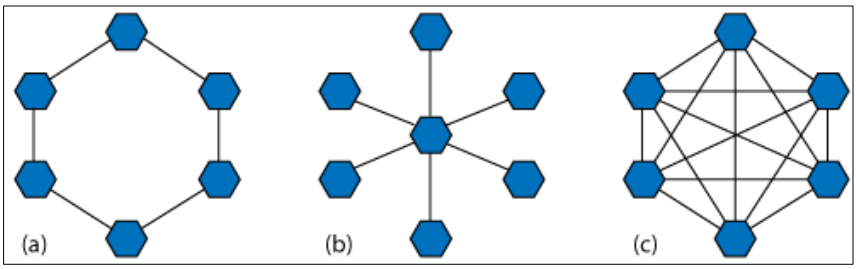


Figure 1-8 Différents types de topologies pour un essaim de particules.

1.3.3.2 Méthode du recuit simulé (SA)

La méthode du recuit simulé (en anglais « Simulated Annealing »), est une méta-heuristique d'exploration stochastique proposée en 1983 par les trois physiciens de la société IBM Kirkpatrick, Gelatt et Vecchi en 1983 [38], et indépendamment par Siarry en 1984 [39] et Cerny en 1985 [40], elle est inspirée d'un processus utilisé en métallurgie pour améliorer la qualité d'un solide.

- **Origine et fonctionnement**

La méthode du recuit simulé est une généralisation de la méthode Monte-Carlo ; son but est de trouver une solution optimale pour un problème donné. On alterne dans cette dernière des cycles de refroidissement lent et de réchauffage (recuit) qui ont pour effet de minimiser l'énergie du matériau. Cette méthode est transposée en optimisation pour trouver les extrema locaux d'une fonction.

Son fonctionnement se présente comme suit [14]: on commence d'abord par chauffer le métal jusqu'à une certaine température où il devient liquide (les atomes peuvent donc circuler librement). Après avoir atteint ce stade, on baisse la température très lentement de sorte à obtenir un solide. Si cette baisse de température est brusque on obtient alors du verre « état solide amorphe » ; si au contraire cette baisse de température est très lente (laissant aux atomes le temps d'atteindre l'équilibre statistique), nous obtiendrons « état solide cristallin » : des structures de plus en plus régulières, jusqu'à atteindre un état d'énergie minimale correspondant à la structure parfaite d'un cristal, on dit alors que le système est « gelé ».

En fait, les thermodynamiciens ont remarqué qu'une baisse brutale de la température d'un liquide entraîne une reproduction d'un optimum local, i.e. une structure amorphe. Alors qu'une baisse progressive de la température du liquide permet d'aboutir à un optimum global, i.e. une structure bien construite. C'est l'idée prise en considération par les métallurgistes qui savent que si le métal refroidit trop vite, il contiendra beaucoup de défauts microscopiques et s'il refroidit lentement ils obtiendront une structure bien ordonnée [41].

L'algorithme SA établi le lien entre ce type de comportement thermodynamique et la recherche de minima globaux pour traiter les problèmes d'optimisation combinatoire.

On dit généralement que l'algorithme SA est le plus ancien de la famille des méta-heuristiques, et certainement l'un des rares algorithmes ayant des stratégies explicites pour éviter les minima locaux. L'idée fondamentale est d'accepter les solutions qui aboutissent à des solutions de qualité inférieure à celle de la solution actuelle afin d'échapper aux minima locaux. La probabilité de faire un tel mouvement est réduite pendant la recherche.

- **Algorithme de SA**

La méta-heuristique du recuit simulé s'inspire de l'algorithme de Métropolis [42], qui permet de décrire l'évolution d'un système thermodynamique, on part d'une configuration donnée, et on lui fait subir une modification aléatoire. Si cette modification fait diminuer la fonction objectif (ou énergie du système), elle est directement acceptée ; Sinon, elle n'est acceptée qu'avec une probabilité égale à $\exp(\Delta E/T)$ (avec E=énergie, et T=température), cette règle est appelée critère de Metropolis [43].

Pour l'algorithme du recuit simulé, on applique itérativement l'algorithme de Metropolis, pour engendrer une séquence de configurations qui tendent vers l'équilibre thermodynamique : commençant par choisir une température de départ T=T0 et une solution initiale S=S0 ; ensuite générer une solution aléatoire dans le voisinage de la solution actuelle ; après comparer les deux solutions selon le critère de Metropolis; répéter les deux dernières opérations jusqu'a ce que l'équilibre statistique soit atteint ; enfin décroitre la température et répéter jusqu'a ce que le système soit gelé (atteinte du critère d'arrêt). La Figure 1–9 représente l'organigramme de l'algorithme du recuit simulé.

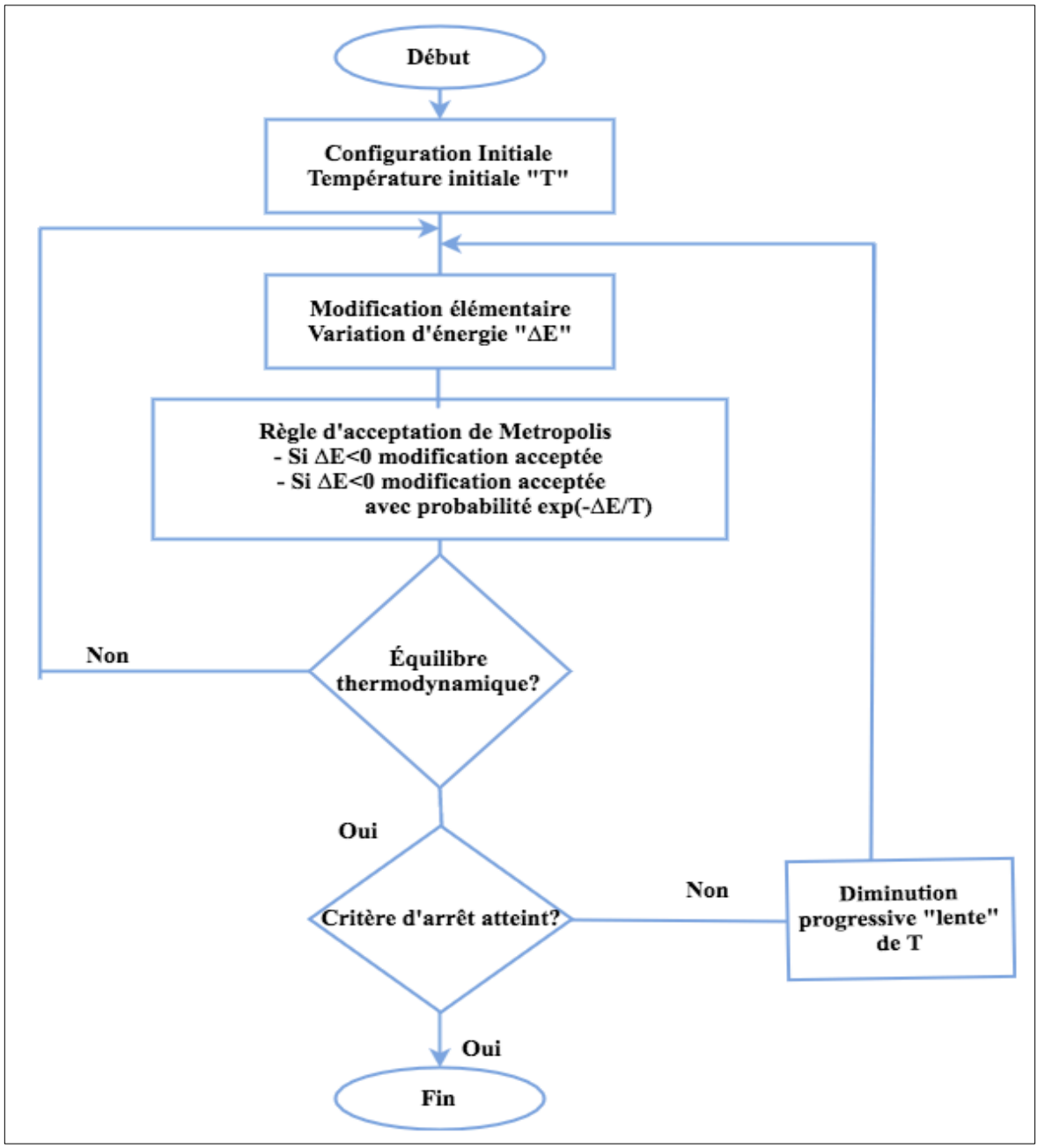


Figure 1-9 Organigramme de l'algorithme SA.

1.4 Conclusion

Ce chapitre est une initialisation aux différentes notions utilisées dans nos travaux de thèse.
Il porte sur le principe général du concept de la parallélisation en informatique, en décrivant un ensemble de ses propriétés, ses différentes stratégies et modèles.
Ce chapitre aborde également le concept de la programmation concurrente, utilisé dans nos modèles parallèles sur la méthode PSO.

Une autre partie a été consacrée au concept d'optimisation en mettant en exergue les deux méta-heuristiques : l'optimisation par essaim particulaire (PSO) méthode approchée à population de solutions et le recuit simulé (SA) méthode approchée à solution unique. Les modèles proposés dans les chapitres qui suivent seront basés sur ces deux méta-heuristiques.

Le prochain chapitre quant à lui aborde la description de deux modèles parallèles pour la méthode PSO ainsi que le développement des résultats obtenus pour les dits modèles.

Chapitre 2Proposition de deux modèles parallèles basés sur la méthode PSO

SOMMAIRE

2.1 Introduction

Pour certains problèmes d'optimisation, les méthodes de résolution dites exactes, ne permettent pas de trouver la solution optimale dans une durée de temps raisonnable. C'est l'une des principales raisons qui ont contribué à la naissance des méta-heuristiques. Cette famille d'algorithmes permet en effet de résoudre des problèmes d'optimisation complexes face auxquels les méthodes classiques manquent d'efficacité. Cependant ces algorithmes de recherche ne peuvent garantir l'optimalité de la solution trouvée.

La méthode PSO a rapidement pris place parmi les méthodes les plus performantes dans la famille des méta-heuristiques dédiée à la résolution des problèmes d'optimisation complexes. Bien que cette méta-heuristique soit très populaire par sa robustesse, elle présente néanmoins plusieurs inconvénients dont les plus étudiés sont le temps de calcul élevé et la convergence prématurée.

Ce chapitre présente deux modèles basés sur PSO et visant à échapper à ces deux inconvénients de la méthode.

Le premier modèle propose une nouvelle notion de voisinage géographique permettant une meilleure exploration de l'espace de recherche dans le but d'améliorer la qualité de la solution et d'éviter la convergence prématurée de la méthode. Le calcul parallèle y est utilisé pour accélérer les calculs et diminuer le temps de calcul afin d'obtenir une solution optimale en un temps de calcul raisonnable.

Le deuxième modèle quant à lui repose sur la division de l'espace de recherche en sous espaces, et le lancement en parallèle d'un ensemble de PSO indépendants dans chaque sous espace à la recherche de l'optimum. Cela permet une meilleure exploration et exploitation de l'espace de recherche en un temps raisonnable.

Une expérimentation des deux modèles a été effectuée. Les résultats obtenus ont été satisfaisants et seront présentés en détails dans la suite de ce chapitre.

2.2 Modèles parallèles basés sur la méthode PSO

Bien que la méthode PSO soit connue pour être l'une des méthodes les plus performantes dans le domaine de l'optimisation, comme toute méta-heuristique elle présente des défauts, la convergence prématurée et le temps de calcul étant les plus traités dans ce domaine.

La convergence prématurée: La structure d'exécution et le principe de déplacement des particules pour la méthode PSO mènent à une convergence prématurée.

A titre d'exemple: si une particule de l'essaim se dirige vers un site " contenant une solution sous-optimale", et se déclare à une itération meilleure de son groupe, cette information affectera le déplacement de toutes les particules de l'essaim lors de la prochaine itération. Les particules vont se diriger vers ce site, ce qui engendra une convergence prématurée.

Le temps de calcul: tous les calculs pour la méthode PSO se font de façon séquentielle, ce qui peut amener à un temps de calcul considérable, surtout pour les problèmes d'optimisation complexes.

Exemple: Si on lance l'algorithme PSO sur une population de particules de "dimension élevée", les calculs se feront à chaque itération pour chaque particule pour toutes ses dimensions de façon séquentielle ce qui provoquera un temps élevé de calcul.

Comme mentionné précédemment, dans l'implémentation de l'algorithme classique de la méthode PSO, tous les calculs se font d'une manière séquentielle, c'est de là que découle l'idée de la parallèlisation afin d'améliorer les performances de l'algorithme. Plusieurs scénarios sont proposés, nous distinguons :

- En **2005,** dans [44], une version parallèle de l'algorithme d'optimisation par essaim de particules est présentée, ainsi que trois stratégies de communication utilisables en fonction de l'indépendance des données. La première stratégie est conçue pour les paramètres de solutions indépendantes ou faiblement corrélées, telles que les fonctions test Rosenbrock et Rastrigrin. La deuxième stratégie de communication peut être appliquée aux paramètres plus fortement corrélés, tels que la fonction Griewank. Dans les cas où les propriétés des paramètres sont inconnues, une troisième stratégie de communication hybride peut être utilisée. Les résultats expérimentaux démontrent l'utilité de l'algorithme parallèle proposé.

- Dans [45], ont testé en **2006** à la fois les algorithmes synchrones et asynchrones des PSO parallèles pour l'optimisation des paramètres typiques des ailes d'un avion de transport. Le résultat indique que l'algorithme PSO asynchrone fonctionne mieux que le PSO synchrone en terme d'efficacité parallèle. [46] ont mis en œuvre en **2006** un algorithme PSO parallèle asynchrone pour les problèmes de tests analytiques et biomécaniques. Les résultats expérimentaux obtenus montrent que le PSO asynchrone est 3,5 fois plus rapide que l'algorithme du PSO synchrone.

- Byung et Alan en **2006** [47] ont effectué des analyses pour déterminer les performances parallèles de la méthode PSO, en évaluant quatre problèmes de test analytique et un problème de test biomécanique. Ils ont calculé les performances parallèles des tests élémentaires sur des clusters Linux homogènes et hétérogènes. Dans le cas asynchrone, ils ont considéré l'un des processeurs comme maître et les autres comme esclaves. Le processeur maître est utilisé pour initialiser tous les paramètres d'optimisation et pour mettre à jour les positions des particules. Le cas asynchrone parallèle utilise la file d'attente First-In-First-Out (FIFO) pour évaluer les particules. La communication entre les processeurs maîtres et esclaves est réalisée à l'aide d'un schéma de communication point à point mis en œuvre avec l'interface de transmission de message. Les résultats obtenus pour les différents modèles parallèles proposés sont meilleurs que ceux du modèle séquentiel.

- Dans [48] en **2007**, une approche PSO parallèle nommée (MRPSO) basée sur le modèle de programmation parallèle MapReduce, dans le but de traiter les problèmes d'optimisation complexes. MRPSO a été appliquée a un ensemble de fonctions test très connues dans le domaine de l'optimisation pour leur difficulté. D'après les résultats obtenus, MRPSO peut gérer jusqu'à 256 processeurs pour des problèmes d'optimisation moyennement difficiles et tolère les pannes de nœuds.

- Dans cette étude [49] effectuée en **2010**, deux algorithmes sont développés pour la détermination de la tarification des options en utilisant l'optimisation par essaim de particules. Le premier algorithme que nous avons développé est l'algorithme synchrone d'évaluation des options utilisant PSO (SPSO), et le second est l'algorithme synchrone parallèle d'évaluation. Les résultats de tarification obtenus de ces deux algorithmes sont proches par rapport au

modèle classique de Black-Scholes-Merton pour les options européennes simples. Un test de l'algorithme PSO parallèle synchrone dans trois architectures a été effectuée sur: une machine à mémoire partagée utilisant OpenMP, une machine à mémoire distribuée utilisant MPI et une architecture multicœur homogène exécutant MPI et OpenMP (modèle hybride). Les résultats montrent que le modèle hybride gère bien la charge lorsqu'il y a une augmentation du nombre de particules en simulation tout en maintenant une précision équivalente.

- Un algorithme parallèle d'optimisation par essaim de particules est décrit dans [4], proposé en **2012** pour résoudre le problème de couverture des jeux de poursuite-évasion, où plusieurs poursuivants doivent coopérer pour couvrir la zone de fuite potentielle d'un fraudeur agile dans un délai raisonnable. La zone à couvrir est complexe et donc difficile à calculer analytiquement. Avec l'utilisation de l'algorithme PSO parallèle proposé, la couverture maximale est atteinte en moins de temps, compte tenu du nombre minimal de poursuivants. Le temps de calcul peut être réduit davantage en optimisant la fonction de fitness en fonction de la localité des données. De plus, l'utilisation d'une longueur variable de trame de données de communication permet de réduire le temps de communication entre processus lorsque le nombre de processeurs augmente (plus de quatre dans l'exemple de test). Les résultats de la simulation montrent une comparaison entre l'accélération, le temps de calcul avant et après l'optimisation de la fonction de fitness et le temps de communication entre trames de données fixes et variables. Les positions et les orientations des poursuivants sont également présentées pour montrer l'efficacité de l'algorithme parallèle proposé.

- Dans [5], les auteurs en **2014** introduisent plusieurs squelettes fonctionnels parallèles qui, dans une implémentation séquentielle de la méthode PSO, en fournissent automatiquement les implémentations parallèles correspondantes. Ils utilisent ces squelettes et rapportent quelques résultats expérimentaux. Ils constatent que, malgré le faible effort requis par les programmeurs pour utiliser ces squelettes, leurs résultats empiriques montrent que les squelettes proposés atteignent des vitesses d'accélération raisonnables.

- En **2015**, Les auteurs de [50], ont développé la problématique des problèmes de satisfaction des contraintes (Constraint Satisfaction Problems CSP) qui se produisent dans

différents domaines. Plusieurs méthodes sont utilisées pour les résoudre. En particulier, la méta-heuristique PSO qui permet de résoudre efficacement les CSP en réduisant considérablement le temps de calcul nécessaire pour explorer l'espace de recherche des solutions. Cependant, PSO est excessivement coûteuse face à de grandes instances. Pour ce travail, un intérêt particulier a été porté aux problèmes de satisfaction de contrainte maximale (Max-CSP) en proposant une nouvelle approche de résolution qui permet de résoudre efficacement les Max-CSP, même avec de grandes instances. L'objectif était d'implémenter une méthode basée sur PSO en utilisant l'architecture GPU (Graphics Processing Unit) en tant que cadre d'informatique parallèle. Deux modèles parallèles sont proposés; le premier est un PSO parallèle GPU pour Max-CSP (GPU-PSO) et le second est un PSO distribué par GPU pour Max-CSP (GPU-DPSO). Les résultats expérimentaux montrent l'efficacité des deux approches proposées et leur capacité à exploiter l'architecture GPU.

- Dans [6] deux stratégies parallèles sont proposées en **2019**, basées sur plusieurs essaims pour résoudre les problèmes d'optimisation multi-objectif. Les multiples essaims co-évoluant en parallèle et interagissant par le biais de la migration. Différentes politiques de déclenchement de la migration sont proposées et évaluées. Une évaluation expérimentale approfondie des algorithmes est présentée, ainsi qu'une étude de l'impact des méthodes proposées sur la convergence et la diversité de la recherche dans de nombreux scénarios de l'optimisation multi-objectifs. La première stratégie est basée sur la domination de Pareto et l'autre sur la décomposition. Plusieurs essaims s'exécutent sur des processeurs indépendants et communiquent en diffusion sur un réseau entièrement connecté. Une étude de l'impact de l'utilisation des stratégies de communication synchrones et asynchrones pour l'approche basée sur la décomposition. Des résultats expérimentaux ont été obtenus pour plusieurs problèmes de référence. La conclusion était que la parallélisation a un effet positif sur la convergence et la diversité du processus d'optimisation pour des problèmes multi-objectifs. Cependant, il n'existe pas de stratégie unique donnant les meilleurs résultats pour toutes les catégories de problèmes. En termes d'évolutivité, pour des nombres d'objectifs plus élevés, les algorithmes parallèles basés sur la décomposition présentent toujours les meilleurs résultats.

2.3 Approches proposées : deux parallélisations basées sur la méthode PSO

Dans cette section, deux modèles parallèles PPSO1 et PPSO2 basés sur la méthode PSO seront développés accompagnés des résultats expérimentaux de chaque modèle.

2.3.1 Proposition du premier modèle parallèle (PPSO1)

Le scénario que nous avons adopté dans ce premier modèle dénommé PPSO1 permet de paralléliser les calculs en lançant un ensemble de threads sur des lots de particules se positionnant dans des voisinages différents.

Les threads, (sorte de processus Java dans notre expérimentation), s'exécutent en parallèle à chaque itération de l'algorithme. Chaque thread exécute le traitement d'une itération de son lot de particules, et attend que les autres threads finissent leurs traitements pour mettre à jour les voisinages et commencer une nouvelle itération. Ce scénario se répète jusqu'à l'obtention d'une solution satisfaisante: "atteinte du critère d'arrêt".

La particularité de ce modèle consiste à profiter de la robustesse de l'algorithme PSO dans le choix du bon paramétrage afin de créer la diversité dans la recherche (dans notre cas: la répartition des particules dans l'espace de recherche et notre notion de voisinage) et dans le partage de l'information pour faciliter la convergence. Le calcul parallèle permet d'accélérer les calculs afin d'avoir une solution "optimale" dans un temps de calcul optimisé.

Figure 2-1 est une représentation de l'approche proposée [51].

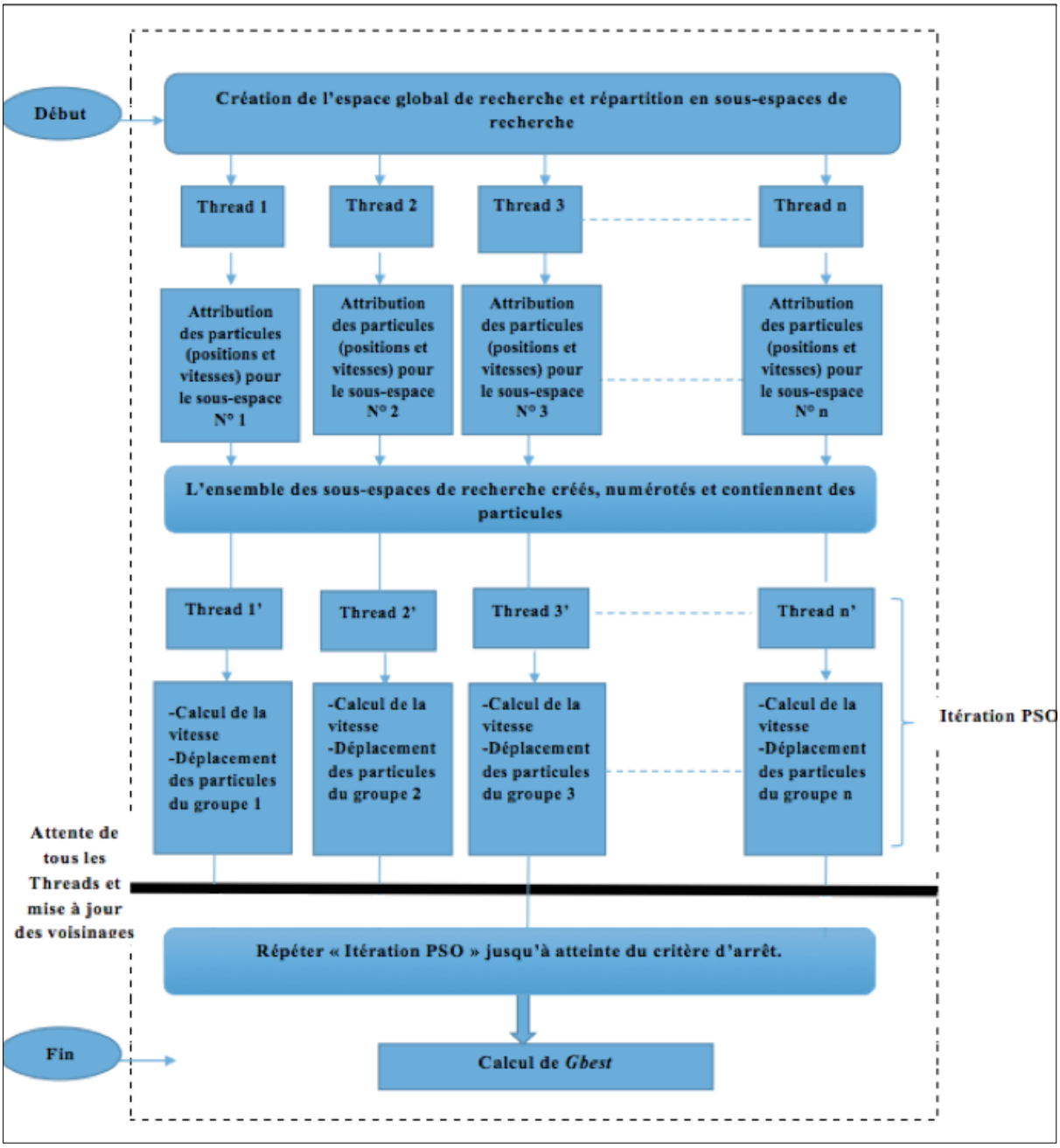


Figure 2-1 Représentation du modèle parallèle proposé "PPSO1".

2.3.1.1 Création des voisinages

Pour le présent modèle PPSO1 basé sur la méthode PSO, la démarche de création des voisinages en deux dimensions est effectuée de la manière suivante:

1. On commence tout d'abord par la création de l'espace global de recherche selon les valeurs du domaine de recherche de la fonction objectif.
2. Ensuite, on divise l'espace global en sous espaces à travers une valeur du "pas" qu'on attribue à chaque axe: les valeurs des pas a et b sont attribuées respectivement aux axes X et Y.
3. On attribue un numéro à chaque voisinage "cellule". L'objectif est qu'à chaque itération, chaque particule doit connaître son voisinage et chaque voisinage doit connaître les particules qui le composent.
4. Pour chaque particule P nous avons ses cordonnées sur l'axe X et l'axe Y respectivement P_X et P_Y.
5. Ensuite on divise P_X par la valeur du pas "a" et P_Y par la valeur du pas "b" pour avoir les valeurs de quotient respectivement k et n.
6. Pour calculer le voisinage $V(P)$ de la particule P, nous allons utiliser la formule mathématique (2-1):

$$V(P) = nl + k + 1 \qquad (2\text{-}2)$$

Avec: $l = X_{\max} / 2$

Tel que: $X_{\max}$ est la valeur maximale de l'axe X.

- **Exemple**

Ci-dessous un exemple explicatif pour une fonction dont :

Le domaine de recherche est: $X \in [0,12]$ et $Y \in [0,6]$.

Pour la particule $P(11,3.5)$, se positionnant dans le voisinage №: 24.

Avec : a=2, b=1.

Et : $X_{\max} = 12$, $Y_{\max} = 6$.

Si on remplace les valeurs dans la formule (2-3), on obtiendra le chiffre "24" qui représente le numéro du voisinage contenant la particule P. (Voir la Figure 2-2). Il est à remarquer que cette approche peut se généraliser à des espaces de dimension supérieure.

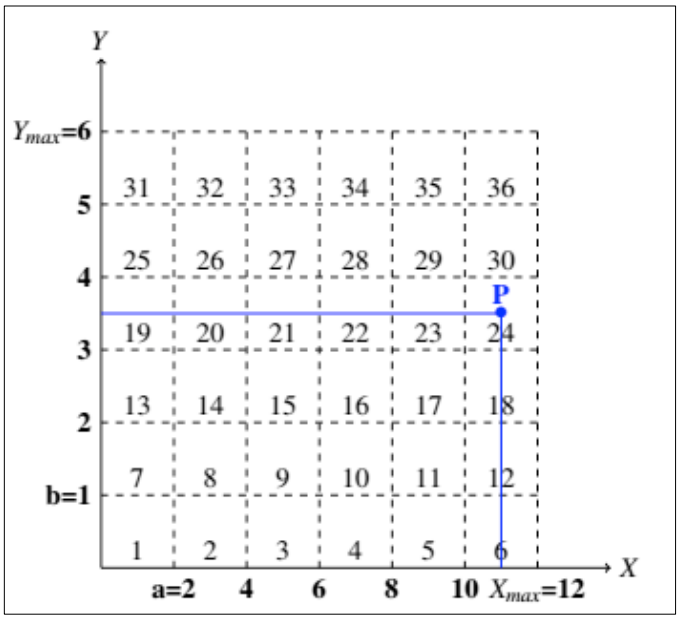


Figure 2-2 Exemple de la représentation des voisinages en deux dimensions.

2.3.1.2 Étapes de l'algorithme parallèle PPSO1 proposé

Les principales étapes de l'algorithme parallèle proposé sont les suivantes:

Étape 1: Créer l'espace global de recherche et le diviser en sous espaces de recherche (comme expliqué ci-dessus).

Étape 2: Attribuer des particules pour chaque sous espace de recherche en générant leurs positions, vitesses et topologie de communication.

Il est à noter que nous avons utilisé le générateur de séquence SOBOL pour l'attribution des positions aux particules, ce dernier permet une génération homogène des particules dans l'ensemble de l'espace de recherche en prenant en considération la forme de l'espace de recherche de chaque problème à optimiser, ce qui va de pair avec notre approche parallèle.

Pour les vitesses, on considère que la taille de l'essaim est constante, et on attribue à chaque particule une vitesse initiale.

Concernant la topologie de communication, nous avons implémenté les trois topologies (étoile, anneau et rayon), mais lors des simulations pour ce modèle, c'est la topologie étoile qui a donné les meilleurs résultats.

Étape 3: Diviser le traitement de l'algorithme PSO sur un ensemble de traitements: pour chaque traitement on attribue un thread.

Étape 4: Attribuer les lots de particules aux threads.

Étape 5: Lancer les traitements de tous les threads en parallèle pour une itération.

Étape 6: Mettre à jour les voisinages (Après chaque déplacement, la particule vérifie si elle a quitté son ancien sous-espace, si oui elle se retire de son ancienne topologie et s'inscrit dans son nouveau sous-espace), et calculer les optimums Pb et Pg.

Étape 7: Si le critère d'arrêt est satisfait, arrêter, sinon passer à l'étape 5.

Étape 8: Le résultat est la meilleure solution obtenue parmi les threads.

Il est à remarquer que l'étape 6 peut également se faire en parallèle sur les voisinages.

2.3.2 Proposition du deuxième modèle parallèle (PPSO2)

Ce second modèle, comme le précédent porte sur la recherche de solutions pour une éventuelle amélioration de l'algorithme PSO: la convergence prématurée et le temps de calcul.

Dans le PSO standard, toutes les particules sont mises à jour directement par leur progéniture, peu importe si elles sont améliorées. Si une particule se déplace vers une meilleure position, lors de son prochain mouvement, elle sera remplacée par cette dernière.

Cependant, si elle se déplace vers une solution considérée "mauvaise", cette nouvelle position impactera son prochain déplacement ainsi que celui des autres particules de l'essaim.

En fait, dans la plupart des cas la majorité des particules se dirige vers des positions sous optimales, donc tout l'essaim convergera vers un optimum local. L'exploitation et l'exploration de l'espace de recherche représentent deux comportements contradictoires, qui fonctionnent ensemble pour résoudre le problème, et le bon équilibre entre eux fait partie intégrante de la performance de l'algorithme PSO. La corrélation de ces comportements avec notre approche est remarquable, cela a été l'inspiration derrière la première partie de notre algorithme. La deuxième partie consiste à minimiser les coûts de calcul en utilisant le calcul parallèle. Notre approche basée sur l'algorithme PSO est décrite dans la Figure 2–3 [52].

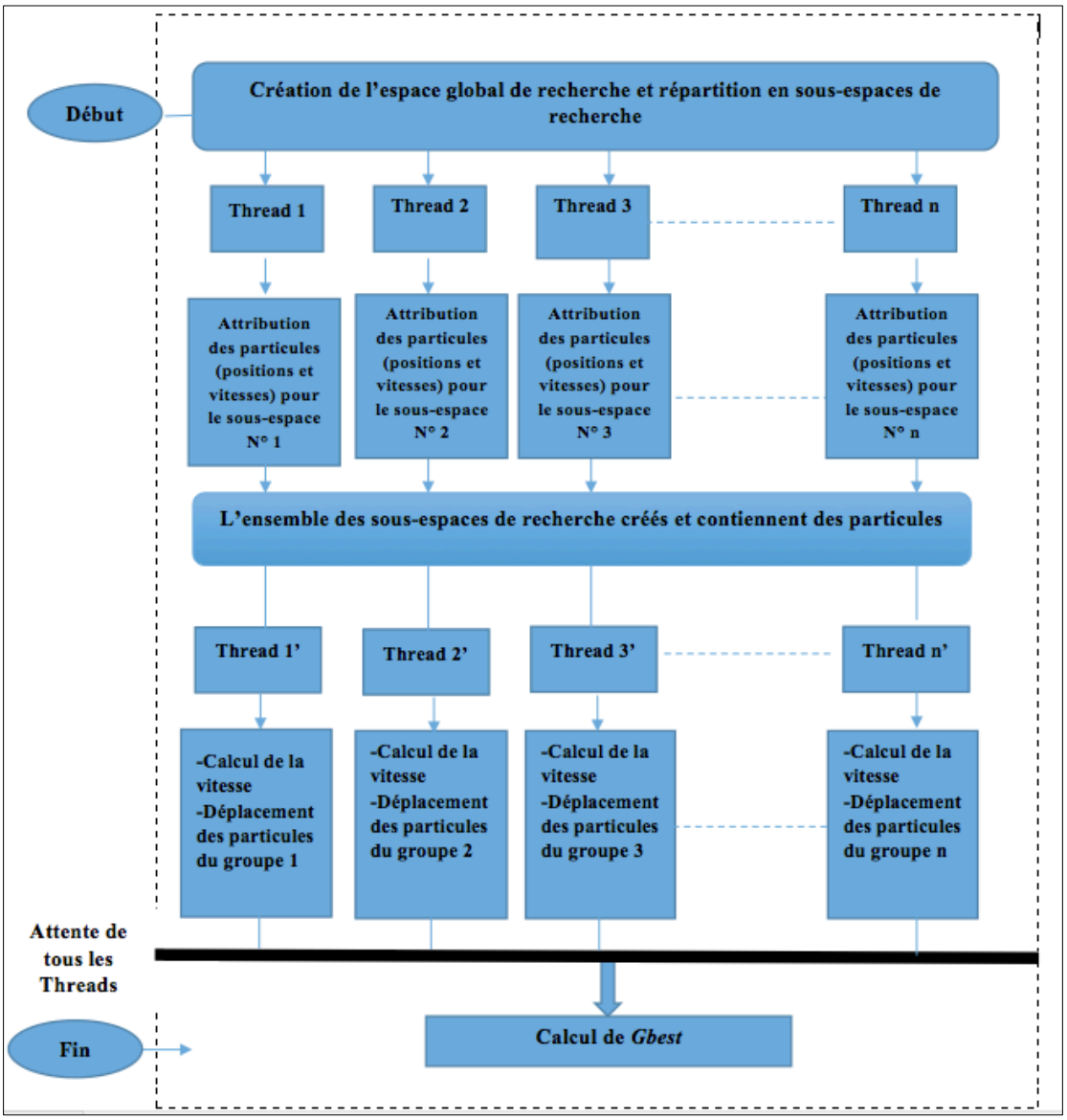


Figure 2-3 Représentation du modèle parallèle proposé "PPSO2".

Dans ce modèle, nous commençons par créer l'espace global de recherche, ensuite on génère les sous-espaces en se basant sur un vecteur de "pas" ou un nombre de "pas" attribué à chaque axe du domaine de recherche. Ensuite, l'attribution des particules aux sous-espaces: initialisation de chaque sous-espace par un ensemble de particules via l'affectation de leurs positions et vitesses initiales.

Ci-dessous une capture d'écran (voir Figure 2–4) de la partie graphique de l'interface utilisateur pour un exemple d'une distribution des particules en deux dimensions: L'espace de recherche global [0,4] est divisé en 16 sous-espaces, à travers une valeur de "pas" égale à 1 dans les deux axes de l'espace de recherche. Chaque sous-espace contient 8 particules de couleur et de forme différentes.

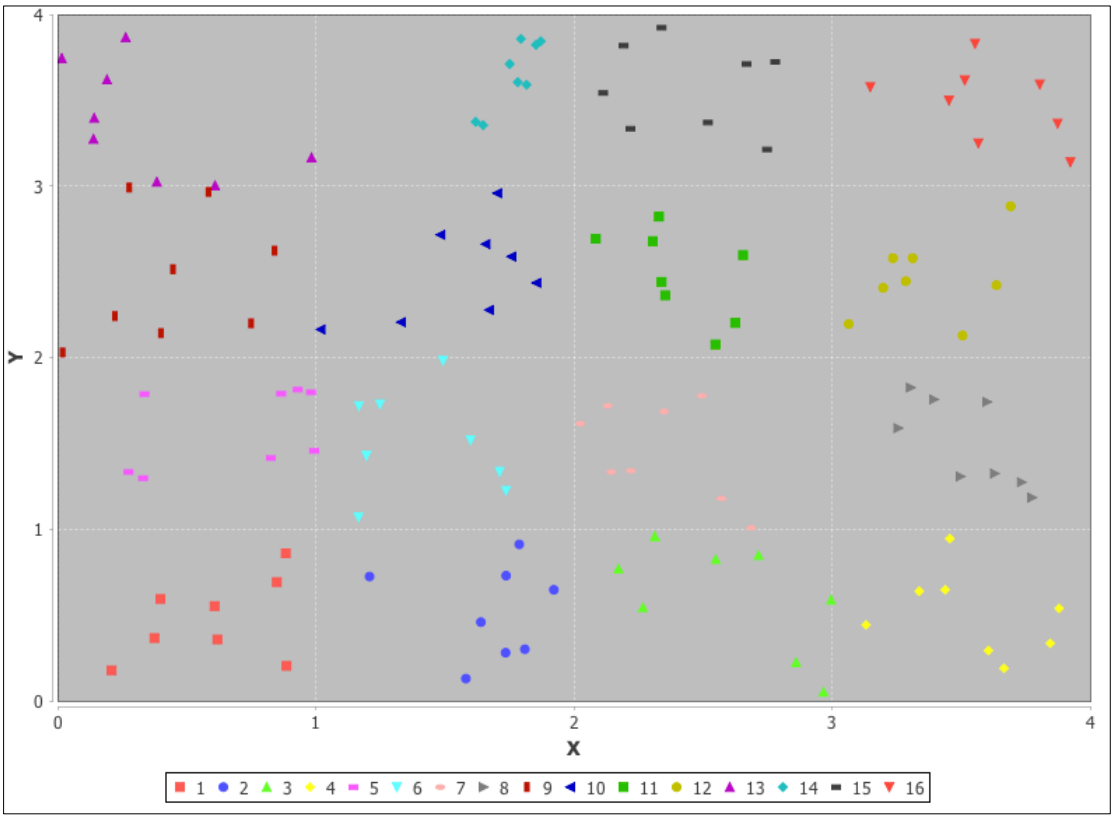


Figure 2-4 Capture d'écran de la partie graphique de l'interface utilisateur.

Ce processus de création des sous-espaces en deux dimensions est généralisé pour des dimensions plus élevées. Après avoir divisé l'espace de recherche global en sous-espaces et les avoir initialisés par des particules, le traitement de chaque sous-espace est attribué à un "thread".

Notre modèle est basé sur le traitement parallèle. En fait, les threads doivent effectuer des calculs sur des lots de particules. Chaque lot de particules est situé dans son sous-espace formant un groupe. Il n'y a pas de communication entre les particules des différents sous-espaces: chaque groupe cherche indépendamment la solution optimale; à la fin, lorsque tous

les threads terminent leur traitement, une comparaison des résultats obtenus pour chaque thread est effectuée afin de trouver la solution la plus optimale.

Ce modèle est conçu pour des problèmes d'optimisation complexes, avec un large espace de recherche et plusieurs optima locaux.

L'objectif principal est de mieux explorer et exploiter l'espace de recherche, non seulement sur la base d'une distribution cohérente des particules dans tout l'espace de recherche, mais également par mouvement de chaque groupe dans son sous-espace pendant tout le processus PSO. La recherche de l'optimum de la fonction objectif s'effectue en même temps pour tous les groupes.

2.3.2.1 Étapes de l'algorithme PPSO2

Les principales étapes de notre algorithme sont les suivantes:

Étape 1: Créer l'espace de recherche global et le diviser en sous-espaces de recherche.

Étape 2: Attribuez des particules à chaque sous-espace de recherche en générant leurs positions, leurs vitesses et leur topologie de communication.

Étape 3: Créer un PSO thread * par sous-espace de recherche.

Étape 4: Évaluez la meilleure solution pour chaque PSO thread. (Une comparaison des résultats obtenus de l'ensemble des groupes et affichage de meilleur résultat obtenu « l'optimum »).

- * Etapes du PSO thread:

Étape 1: Évaluez la fitness de chaque particule.

Etape 2: Pour chaque particule, si sa fitness est plus petite que sa meilleure précédente fitness (Pb), mettre à jour Pb.

Etape 3: Pour chaque particule, si sa fitness est plus petite que la meilleure de toutes les particules (Pg), mettre à jour Pg.

Étape 4: Déplacer toutes les particules selon les formules (1) et (2).

Étape 5: Si le critère d'arrêt est satisfait, arrêter, sinon passez à l'étape 1.

2.4 Description des expérimentations et résultats

La modification de l'algorithme de base de la méthode PSO pour nos deux modèles concerne trois points essentiels: la notion de voisinage, l'adaptation des paramètres, et le calcul parallèle. Ces modifications de l'algorithme améliorent sa performance.

Notre algorithme a été programmé dans JAVA 1.8, et les expérimentations ont été faites sur une machine MacBook Pro OS X 10.13.15, Core i7, 16 Go.

Les threads représentent la technologie utilisée en Java pour rendre les applications multitâches. Ils partagent la même mémoire, ainsi que des ressources (en mémoire), pour ceci, les threads risquent de rentrer en concurrence et de corrompre le système. C'est alors qu'intervient la programmation concurrente qui rassemble un ensemble de fonctionnalités et de techniques pour permettre la synchronisation de tâches s'exécutant en parallèle. Java gère mieux les processus que les threads, mais les threads sont beaucoup plus utilisés parce qu'ils sont mieux intégrés au langage Java et moins gourmands en ressources mémoires.

Certes Java est un langage robuste possédant beaucoup d'avantages (portabilité, réutilisation de classes, l'héritage, etc), mais c'est surtout pour la notion de la programmation concurrente (parallélisme et synchronisation) que nous avons opté pour ce langage afin d'effectuer nos expérimentations et profiter des avantages du parallélisme en terme de diminution du temps de calcul, et d'exploitation maximale des ressources matérielles de la machine.

2.4.1 Fonctions tests

La qualité des méthodes d'optimisation est fréquemment évaluée en utilisant des fonctions de test standard. Cette série de problèmes créés spécifiquement pour tester les performances des algorithmes d'optimisation, telles que: la vitesse de la convergence, la qualité de la solution, la précision, la robustesse, la performance générale, etc.

Dans la littérature, nous avons quatre classes de fonctions de test regroupées en classes toutes continues comme suit [53]:

- (A): Unimodale, convexe, de dimension élevée,
- (B): Multimodale, à deux dimensions avec un faible nombre d'extrema locaux,
- (C): Multimodale, à deux dimensions avec un nombre élevé d'extrema locaux,

- (D): Multimodale, de dimension élevée, avec un nombre élevé d'extrema locaux.

Dix fonctions de test (f1-f10) Tableau 2-1 sont utilisées dans cette simulation des classes B et C. D'une manière générale, les fonctions multimodales sont souvent considérées comme les plus difficiles dans le domaine de l'optimisation car elles possèdent des propriétés similaires aux problèmes du monde réel et fournissent une bonne base pour tester la crédibilité d'un algorithme d'optimisation, notamment du fait de leur grand nombre d'optima locaux. Dans nos expérimentations nous avons opté pour des problèmes à deux dimensions.

Tableau 2-1 Descriptif des fonctions utilisées dans nos expérimentations.

Fonction	Domaine	$f_{\min}$
f_1 Rosenbrock	±30	0
f_2 Himmelblau	±30	-3.78396
f_3 Beale's	±4.5	0
f_4 Easom	±100	-1
f_5 McCormick	±4.0	-1.9133
f_6 Three-hump camel	±5.0	0
f_7 Hölder table	±10	-19.2085
f_8 Matyas	±10	0
f_9 Booth's	±10	0
f_{10} Goldsteinprice	±2.0	3

2.4.2 Paramètres expérimentaux

Dans l'algorithme PSO chaque paramètre a une influence importante sur le comportement des particules et donc de la convergence de l'algorithme; et même si la méthode PSO présente des résultats satisfaisants, le choix du bon paramétrage de la méthode reste un point critique ainsi qu'une des clés de succès pour tout algorithme PSO. Dans la section descriptive de la méthode PSO, nous avons présenté quelques paramètres qui influencent le comportement des particules dans leurs déplacements à la recherche de l'optimum. Le jeu

de paramètres que nous avons élaboré dans notre modèle consiste en l'utilisation de plusieurs paramètres variables, que l'on peut modifier depuis l'interface utilisateur dédiée pour cela. Tout dépend des exigences du problème d'optimisation posé. Une expérimentation massive a été effectuée, pour trouver le jeu de paramètres adéquat ; elle a donné des résultats que nous estimons satisfaisants.

Il est important de noter, qu'un simple changement de la valeur d'un paramètre peut changer grandement le résultat, et peut même mener à une convergence prématurée.

Pour la présente étude qui traite des problèmes de taille modérée, à deux dimensions, la liste des paramètres qui ont donné suffisamment de bons résultats sont mentionnés dans le tableau ci-dessous (voir Tableau 2-2).

Tableau 2-2 Descriptif des paramètres PSO utilisés dans nos expérimentations.

Paramètres PSO	Modèle PPSO1	Modèle PPSO2
Nombre de particules	30	30
Nombre des itérations	50-80	50-80
Coefficients d'accélération	C1 = 1.25, C2=2.25, C3=2.25	C1 = 1.25, C2=2.25
Facteur d'inertie	(0.4 – 0.2)	(0.4 – 0.2)
Topologie de communication	Anneau	Anneau
Nombre de threads	Dépend de la fonction objectif	Dépend de la fonction objectif

Les paramètres de l'algorithme pour les deux modèles PPSO1 et PPSO2 sont définis en fonction de nos approches parallèles. Par exemple: la valeur du facteur d'inertie est variable et plus petite pour une plus grande capacité de recherche locale. La même chose pour la topologie de communication, l'anneau est une topologie adéquate pour une plus grande exploration. Pour les deux modèles, le nombre de threads utilisés dépend de la répartition de

l'espace de recherche. Pour le premier modèle le nombre de threads alloués aux groupes de particules se situant dans des voisinages différents est fixé à l'initialisation de l'algorithme (généralement on alloue le traitement de 5 à 7 particules à un thread). Pour le deuxième modèle, le nombre de threads créés est égal au nombre de zones de recherche, et il est fixé à l'initialisation du programme.

2.4.3 Résultats

Pour le premier modèle PPSO1, les expérimentations réalisées reposent sur le lancement de traitements PSO parallèles sur des lots de particules à la recherche de l'optimum "minimum" de la fonction objectif et en tenant compte des voisinages. La parallèlisation sur les lots de particules s'est avérée judicieuse. Celle concernant les voisinages paraissait coûteuse (calculs inutiles sur des voisinages vides) et a été mise de côté. Le tableau ci-dessous représente le détail de la moyenne des résultats de 1000 exécutions, à savoir les valeurs du temps d'exécution en millisecondes, le SR (Success Rate): le taux de succès qui est le pourcentage de convergence de la fonction vers la bonne solution, et le EvalIF qui représente le nombre d'évaluation de la fonction objectif, et ce pour le modèle séquentiel et parallèle du programme proposé sur un ensemble de dix fonctions (voir Tableau 2-3).

Tableau 2-3 Comparaison des résultats obtenus de PSO et PPSO1.

	PPSO1 parallèle			PSO séquentiel		
Fontions	SR %	Time (ms)	EvalF	SR %	Time (ms)	EvalF
Rosenbrock	100	1007.8	420153	99	1010.1	420209.3
Himmelblau	100	16.9	10032 .1	97	18.2	12019.5
Beale's	100	608.4	25590.2	100	699.3	25799.4
Easom	100	11.1	6997.7	100	11.9	7101.2
McCormick	100	814.9	29531.8	100	891.7	29771.4
Three-hump camel	100	901.5	300501	100	989.8	378721
Hölder table	100	13.9	8945	100	14.5	9000.1
Matyas	100	409.2	134731.2	100	478.9	138077
Booth's	100	11.2	6133.5	100	12.2	6990
Goldstein-price	100	15.5	10012	100	16.1	10928

D'après les résultats obtenus, nous pouvons dire que PPSO1 permet d'obtenir la solution optimale avec une probabilité plus élevée, ainsi que le temps de calcul dans PPSO1 est légèrement plus faible que celui du modèle PSO séquentiel.

Pour le deuxième modèle PPSO2, les expérimentations concernent le lancement d'un ensemble de threads à la recherche de l'optimum "minimum" de la fonction objectif, chaque thread s'occupe du traitement PSO pour son voisinage (les particules de sa zone), la recherche de l'optimum s'effectue en parallèle, il n'y a pas de communication entre les différents groupes. Chaque groupe explore sa zone de recherche indépendamment, et ne communique pas ses résultats à chaque itération ; a la fin du programme, une comparaison des résultats obtenus pour chaque thread est effectuée afin de connaître le meilleur global (voir Tableau 2-4).

Tableau 2-4 Comparaison des résultats obtenus de PSO et PPSO2.

	PPSO2 parallèle			PSO séquentiel		
Fontions	SR %	Time (ms)	EvalF	SR %	Time (ms)	EvalF
Rosenbrock	100	891.2	300123	99	967.1	301213.1
Himmelblau	100	15.4	9702 .5	95	23.7	11089.3
Beale's	100	511.3	21760.2	100	702.4	23099.4
Easom	100	10.3	6222.2	100	12.1	7680.1
McCormick	100	702.9	26201.3	100	911.4	28901.3
Three-hump camel	100	831.6	279008	100	997.9	384098
Hölder table	100	12.8	7907.1	100	14.3	8709.8
Matyas	100	345.5	120312	100	456.7	133098
Booth's	100	9.8	5780.3	100	11.6	7002
Goldstein-price	100	13.4	8872	100	15.6	9995.4

D'après les résultats obtenus, nous avons constaté la performance de PPSO2 en terme de la qualité de la solution, il évite la convergence des particules dans des optimums locaux. Le temps de calcul dans PPSO2 est aussi plus faible que celui du modèle séquentiel.

2.4.4 Analyse des résultats

A partir des différentes simulations réalisées, nous pouvons déduire que :

- Le modèle PPSO1 se basant sur un voisinage géographique dynamique présente des résultats intéressants en terme de la qualité de la solution, mais reste coûteux en terme de temps de calcul, même avec sa parallèlisation, à cause de la synchronisation qui s'effectue à chaque itération.
- Pour le modèle PPSO2, basé sur un voisinage statique, ce dernier ne contient pas de synchronisation, seule une comparaison des résultats obtenus pour chaque thread est effectuée à la fin du programme pour connaître le meilleur global. Cependant, le choix

du critère d'arrêt ainsi que la valeur du « pas » sont très importants : le critère d'arrêt doit être adéquat pour tous les groupes : pas pénalisant en terme de temps de calcul, la valeur du «pas» doit être étudiée parce que tout voisinage créé correspond à un traitement PSO ajouté.

- Le nombre des itérations doit assurer la convergence au résultat, il est donc important de bien l'estimer (fonction du problème à optimiser) car il est à l'origine des temps de calculs plus ou moins important (à priori débuter avec une valeur faible car cela peut être suffisant).
- Le nombre de particules composant l'essaim est lié à la dimension du problème à optimiser. Plus on augmente la dimension du problème, plus on a besoin de particules pour assurer l'optimisation.
- La création des threads étant onéreuse elle doit être étudiée, et cela ne passe qu'à travers une expérimentation massive en modifiant à chaque fois le nombre des sous espaces jusqu'à l'obtention de la valeur adéquate qui est relative à chaque problématique.
- PPSO1 est mieux adapté pour les problèmes d'optimisation de taille moyenne, puisque la synchronisation est coûteuse en terme de temps de calcul.
- PPSO2 est coûteux une fois utilisé pour de petits problèmes d'optimisation, mais reste performant pour les problèmes complexes à plusieurs optimums locaux.

2.5 Conclusion

Ce chapitre présente deux modèles basés sur la méta-heuristique d'optimisation par essaim particulaire. L'objectif était de proposer des solutions aux deux inconvénients de la méthode : la convergence prématurée et le temps de calcul élevé. Dans la littérature, plusieurs versions améliorantes de la méthode PSO sont proposées soit en ajoutant de nouveaux paramètres, en la parallélisant ou en l'hybridant avec d'autres méta-heuristiques.

Les deux modèles que nous avons présenté dans ce chapitre sont basés sur deux concepts : la parallèlisation et le voisinage. La combinaison de ces deux notions a amélioré la performance de la méthode en termes de qualité de la solution et de temps de calcul.

Notre premier modèle PPSO1 utilise la notion de voisinage dynamique, qui permet de créer la diversité dans la recherche ainsi qu'une meilleure exploration de l'espace de recherche afin d'améliorer la qualité de la solution et d'éviter la stagnation de l'algorithme dans un optimum local ; le calcul parallèle est utilisé pour accélérer les calculs dans le but d'avoir une solution "optimale" en un temps de calcul réduit.

Le deuxième modèle PPSO2 repose sur le lancement de plusieurs PSO parallèles: chaque groupe explore indépendamment sa zone à la recherche de l'optimum de la fonction sans être influencé par les résultats des autres groupes, tous les groupes cherchent en parallèle et à la fin une comparaison des résultats obtenus est effectuée pour connaître le meilleur global.

Les deux modèles ont été implémentés en Java et testés sur 10 fonctions test sans contraintes. Les résultats obtenus ont prouvé l'efficacité de nos modèles parallèles par rapport au PSO séquentiel.

La réalisation de ces deux modèles et leurs expérimentations nous a mené à une réflexion plus poussée, afin de réaliser un modèle parallèle plus performant et le tester sur un problème réel d'optimisation, que présentons dans notre prochain chapitre.

Chapitre 3Application d'un modèle parallèle au problème de transport d'électricité

SOMMAIRE

3.1 Introduction

Ce chapitre traite le problème du transport d'électricité ; plus précisément l'optimisation de la durée de vie du pylône d'une ligne de transport d'électricité. L'objectif est de maximiser la résistance à la charge tout en réduisant le coût « minimisation de l'utilisation des matériaux ».

La résolution du problème est basée sur un algorithme parallèle « MPSO », utilisant les threads Java pour le calcul parallèle, ainsi qu'une notion de voisinages évolutifs pour éviter la convergence prématurée de la méthode PSO.
Le modèle MPSO est conçu suite aux observations relatives aux expérimentations des deux modèles PPSO1 et PPSO2 dans le but d'améliorer la qualité des résultats obtenus pour ces deux derniers et d'éviter leurs inconvénients.

Dans nos expérimentations sur la problématique du transport d'électricité les tests effectués sur le programme ont donné des résultats satisfaisants du modèle parallèle MPSO proposé par rapport au processus d'optimisation de premier ordre du logiciel ANSYS.

3.2 Description de la problématique

La production, le transport et la distribution de l'électricité induisent des pertes; qui sont dues à plusieurs raisons. Pour notre étude on effectue une analyse structurale du pylône d'une ligne de transport d'électricité qu'on assimile à un treillis plan [54]. (voir Figure 3-1).

Deux charges identiques F de 1,8 KN sont appliquées aux deux extrémités supérieures du pylône suivant un angle de θ=15°. Les barres formant la structure sont en acier dont le module d'élasticité EX=200 GPa et le coefficient de Poisson PRXY=0.3. La section de chaque barre vaut A=27.90 cm^2. On fait l'hypothèse que le poids de chacune des barres du treillis est négligeable devant les efforts appliqués.

Le problème d'application étudié est la minimisation du poids des barres du treillis, en recherchant des zones transversales optimales. Le schéma statique du problème est présenté à la Figure 3-1. Les positions des nœuds et les connexions des barres sont fixes, seules les sections font l'objet de l'optimisation.

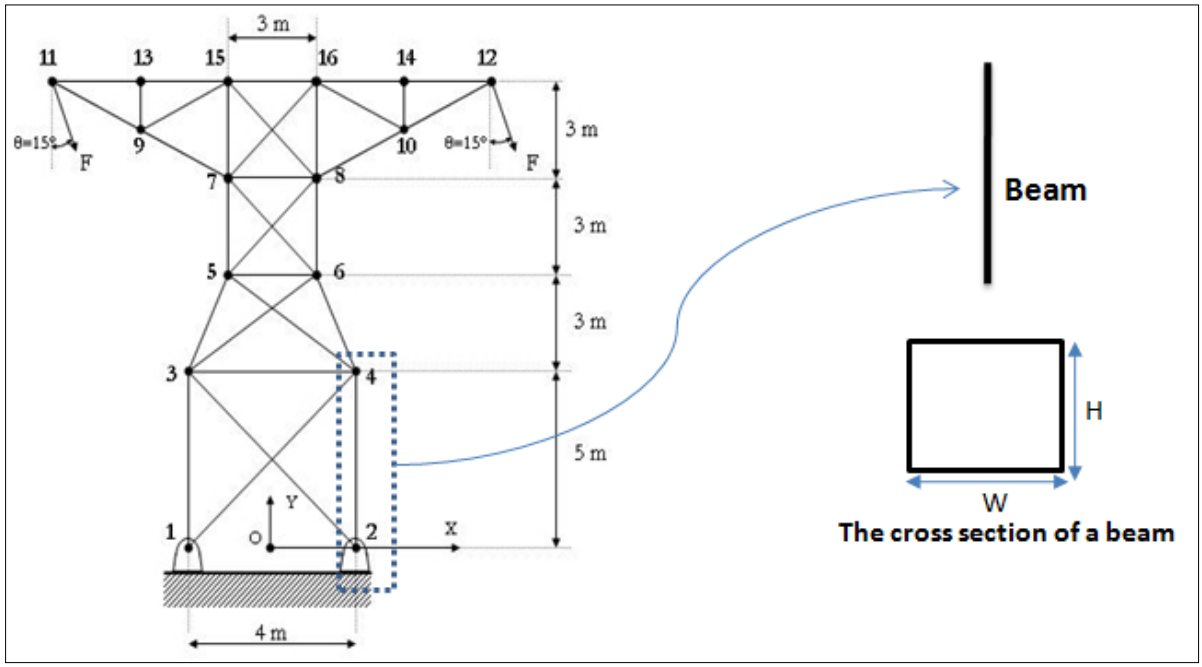


Figure 3-1 Géometrie du pylône électrique

3.3 Description de l'algorithme MPSO proposé

PSO est une méta-heuristique basée sur une population de solutions qui s'est révélé être l'un des algorithmes les plus efficaces inspirée de la nature et destiné à traiter des problèmes d'optimisation globale avec ou sans contraintes. Mais la convergence globale n'est pas assurée à tous les coups avec l'algorithme PSO en raison de la contrainte des particules de rester dans un espace d'échantillonnage fini.

De ce fait, il peut entraîner une convergence prématurée en diminuant la capacité de recherche globale de l'algorithme. D'autre part, la méthode PSO nécessite un temps de calcul considérable surtout pour les problèmes d'optimisation complexes.

Dans le but de résoudre la problématique décrite ci-dessus, un algorithme parallèle basée sur la méta-heuristique PSO est proposé.

Comme mentionné dans le chapitre précédent, l'algorithme PSO de base repose sur un ensemble de traitements qui se font de manière séquentielle. Cependant la performance de la méthode est impactée, surtout pour les problèmes d'optimisation complexes nécessitant un temps de calcul élevé.

Le modèle proposé Modified PSO "MPSO" repose sur une nouvelle notion de voisinage évolutif qui permet de créer la diversité dans la recherche pour mieux explorer l'espace de recherche dans le but d'éviter la convergence prematurée de la méthode et d'obtenir une solution optimale.

L'utilisation des voisinages permet d'obtenir une solution optimale mais reste coûteuse en terme de temps de calcul, d'où l'utilisation du calcul parallèle afin d'accélerer les calculs et d'obtenir une solution optimale en un temps de calcul restreint.

Nos voisinages ont la forme de sphères, qui sont mises à jour à chaque itération: leurs centres évoluent et la valeur du rayon change suivant des conditions relatives au nombre de voisinages.

Tout le traitement PSO est parallélisé, les threads sont crées et lancés à chaque itération, chaque thread s'occupe du traitement PSO de son groupe de particules.

La particularité de notre modèle concerne trois grands axes: le parametrage de l'algorithme PSO selon notre modèle proposé, la notion de voisinage géographique "dynamique", qui

permet de créer la diversité dans la recherche et dans le partage de l'information pour une convergence plus optimale. Aussi bien le calcul parallèle permettant d'accélerer les calculs afin d'avoir une solution optimale dans un temps de calcul optimisé.

La Figure 3-2 est une représentation de l'approche proposée [55].

Le principe de création des voisinages, le traitement parallèle, le critère d'arrêt et l'algorithme vont être developpés ci-dessous.

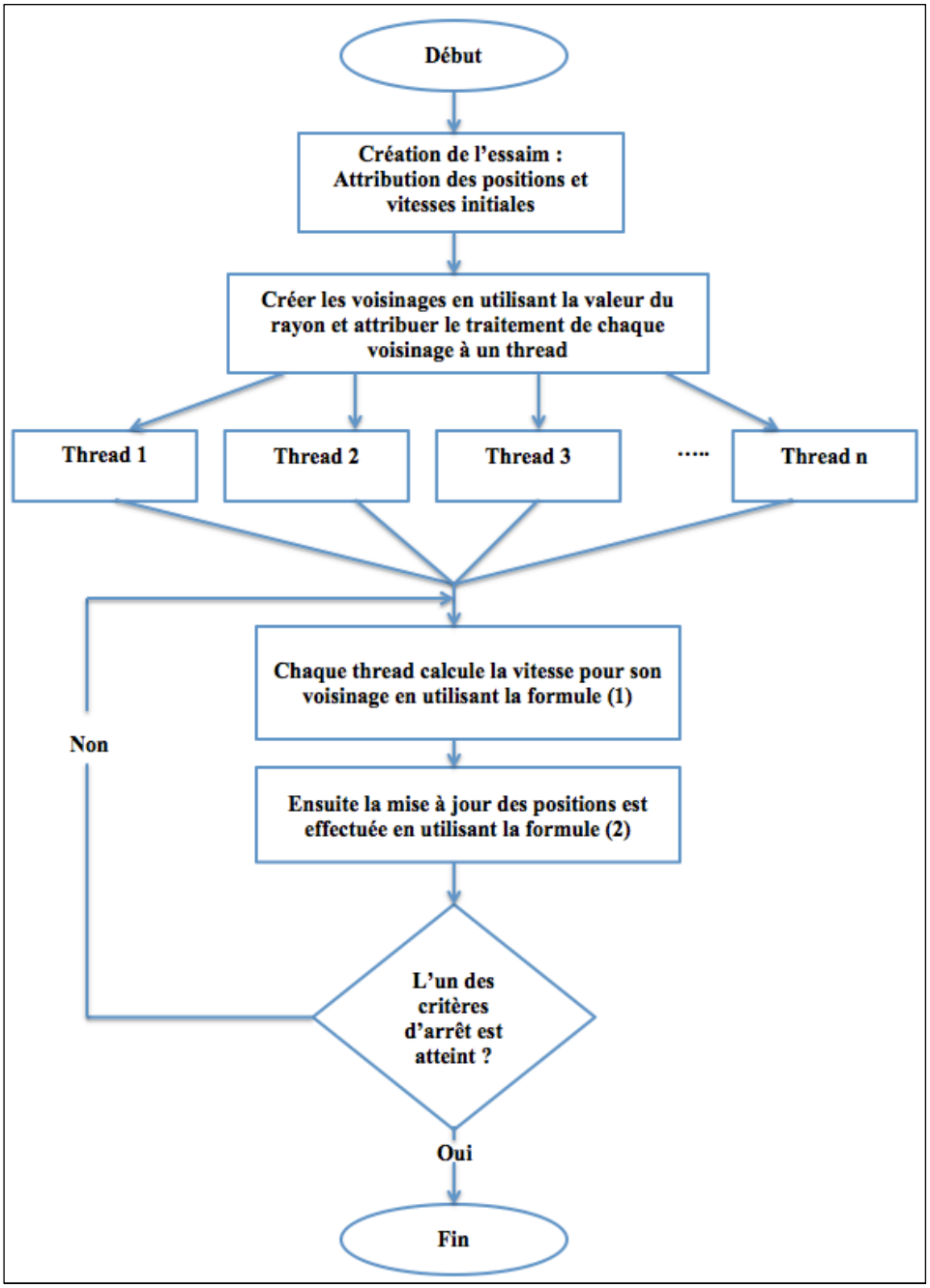


Figure 3-2 Représentation de l'approche MPSO implémentée.

3.3.1 Création des voisinages

Les voisinages sont des sphères dynamiques, à chaque itération le nombre de particules comprises dans les différentes sphères change selon les nouvelles positions des particules et la valeur du rayon.

La création des sphères se fait de la manière suivante: On initialise les positions des particules, on spécifie la valeur initiale du rayon, on considère une première particule Pc. Elle représente alors le centre de la sphère S de rayon r. Une particule Pa est voisine de la particule Pc si la distance euclidienne de Pa à Pc est inférieure ou égale à la valeur du rayon r. Sinon, elle devient centre d'une nouvelle sphère.

Toute nouvelle particule a son appartenance étudiée par rapport aux diverses sphères créées avant de décider de la création d'une nouvelle sphère. Par ailleurs, si le nombre de sphère est réduit (nombre prédéfini) alors le rayon commun aux sphères est diminué sensiblement (voir Figure 3-3 et Figure 3-4).

La particularité des voisinages de notre modèle est qu'on bénéficie des avantages de la notion de voisinage dans le partage de l'information et la coopération entre les sous-essaims, sans avoir à tomber dans le piège d'une convergence prématurée. Dans le modèle classique de l'algorithme PSO avec voisinage, le partage de Pn "la meilleure de chaque voisinage" se fait à chaque itération, et sur la base d'une comparaison de toutes les Pn obtenues, on définit la meilleure de tout l'essaim Pg. Ceci dit, si une particule d'un voisinage se dirige vers un site promoteur (contenant une bonne solution), et qu'elle se déclare meilleure de son voisinage, à la fin de l'itération l'information sera propagée, et cette particule sera déclarée meilleure de tout l'essaim, donc elle influencera la formule de déplacement de toutes les particules, qui vont se diriger vers ce site. Nous supposons que ce site contient un optimum local, et qu'il existe bien une solution optimale quelque part dans l'espace global de recherche, mais vue l'influence de l'information propagée à chaque itération dans le déplacement des particules, ces dernières se dirigent vers le mauvais chemin, ce qui mène à une convergence prématurée.

Ce que nous proposons dans notre modèle, est que les différents voisinages cherchent la solution indépendamment de la valeur du Gb. Chaque particule se déplace en fonction de sa meilleure valeur Pb, et la meilleure de son voisinage Pn.

Notre modèle MPSO respecte toujours le principe de base de l'algorithme PSO basé sur la coopération entre les particules, et le partage de l'information qui existent toujours, puisque les voisinages sont dynamiques.

A chaque itération les particules changent de voisinages et donc elles diffusent leurs informations dans les nouveaux voisinages. Le non partage de la Pg (meilleure position connue globale) à chaque itération permet une meilleure exploitation de l'espace de recherche et donne plus de chance aux particules d'éviter l'anomalie de la convergence prematurée de l'algorithme.

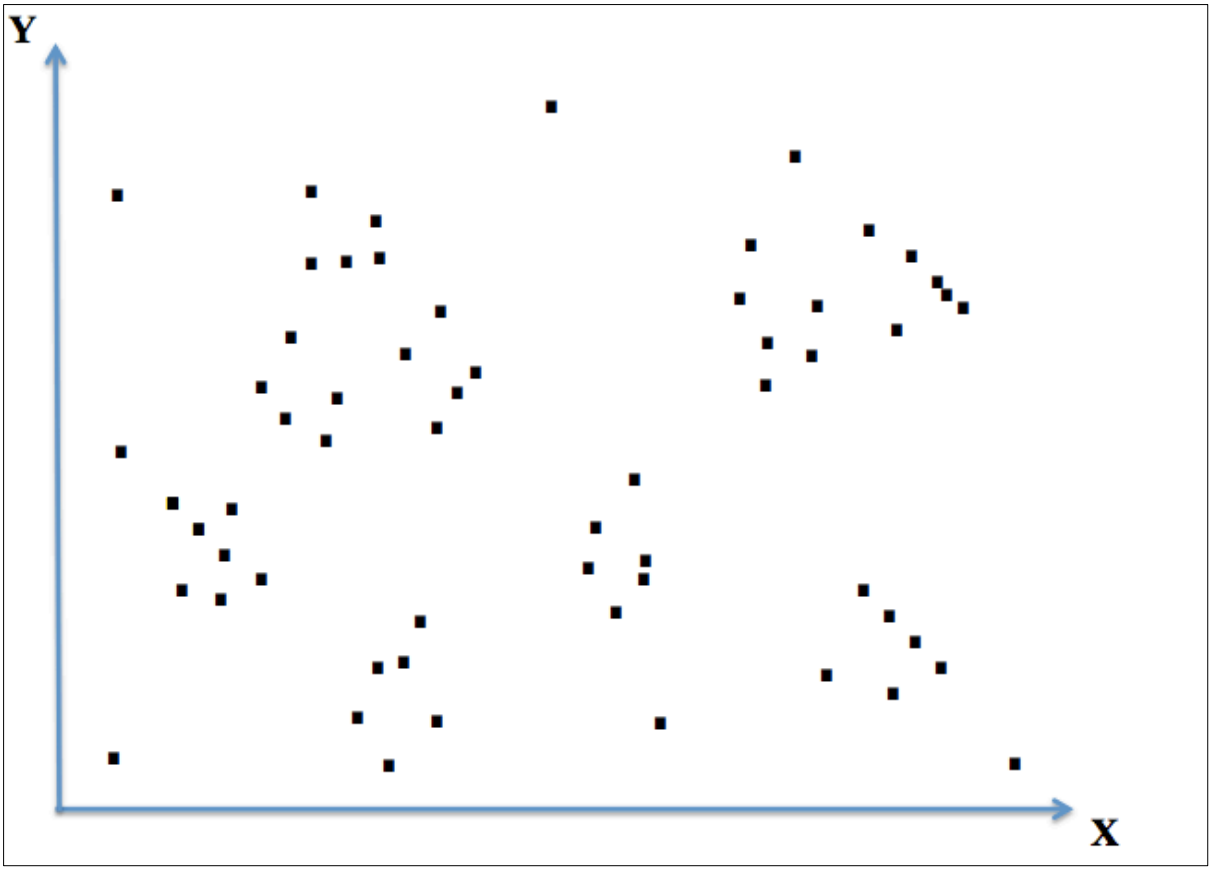


Figure 3-1 Distribution des particules dans l'espace de recherche en deux dimensions.

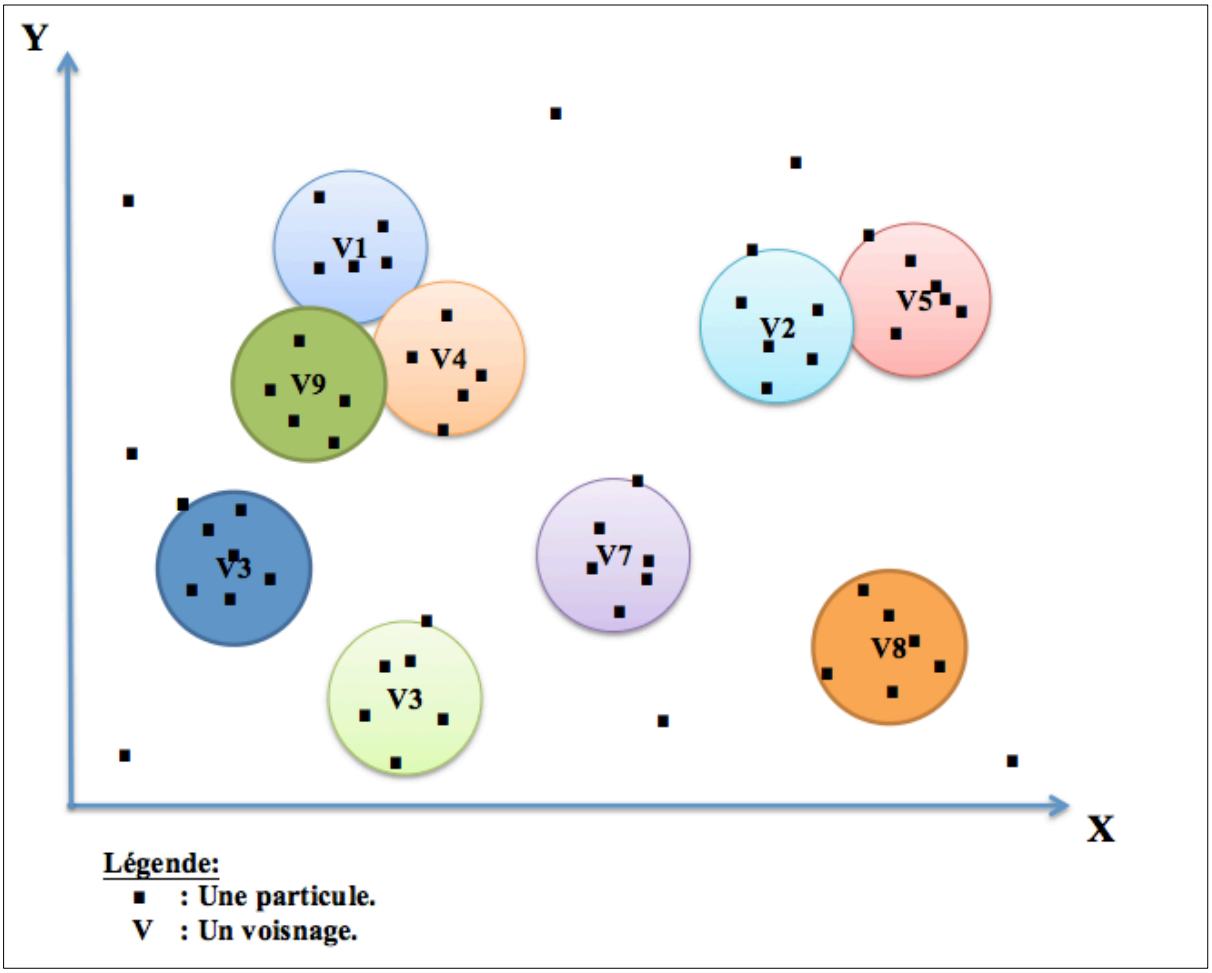


Figure 3-2 Représentation graphique en deux dimensions des voisinages.

3.3.2 Calcul parallèle

Notre approche basée sur l'algorithme PSO, consiste à lancer un ensemble de traitements (threads Java) en parallèle. Chaque thread s'occupe du traitement PSO (évalution des positions et déplacement des particules) d'un lot de particules (composant un voisinage) pour une itération.

À la fin de chaque itération, une synchronisation des threads s'effectue afin d'évaluer les résultats obtenus pour chaque voisinage et mettre à jour les voisinages afin de commencer une nouvelle itération en attribuant le traitement des nouveaux voisinages crées aux threads; ce processus se répète jusqu'à atteinte du critère d'arrêt.

Il est à noter que le nombre des threads à chaque itération est équivalent au nombre de voisinages crées.

3.3.3 Étapes de l'algorithme

Les principales étapes de l'algorithme de notre approche MPSO proposée sont les suivantes:

- **Étape 1:** Attribuer les particules pour l'ensemble de l'espace de recherche en générant leurs positions, vitesses et topologie de communication.
- **Étape 2:** Créer les voisinages via la valeur initiale du rayon.
- **Étape 3:** Diviser le traitement de l'algorithme PSO sur un ensemble de traitements: pour chaque traitement on attribue un thread.
- **Étape 4:** Attribuer les lots de particules aux threads.
- **Étape 5:** Lancer les traitements de tous les threads en parallèle pour une itération.
- **Étape 6:** Mettre à jour les voisinages selon les nouvelles positions des particules et la nouvelle valeur du rayon s'il y a lieu.
- **Étape 7:** Si le critère d'arrêt est satisfait, arrêter, sinon passer à l'étape 5.
- **Étape 8:** Le résultat est la meilleure solution obtenue parmi les threads.

3.3.4 Critère d'arrêt

Le choix d'un critère d'arrêt optimisé n'est généralement pas simple; mais présente l'une des clés de succès de l'algorithme. Pour plus de diversité dans le programme, nous avons opté pour les trois critères d'arrêt suivants:

1. Un nombre maximum d'itérations sans amélioration: nous spécifions un nombre d'itérations au bout duquel sans amélioration remarquable de la solution optimale, le programme s'arrête.
2. Une précision relative au rayon, représentée par une valeur précisant le rayon minimal accepté, si cette valeur est atteinte alors l'exécution du programme s'arrête.
3. Une précision relative à la distance du Gbest, ce critère d'arrêt concerne la valeur de la meilleure position de tout l'essaim, si la distance entre la valeur de Gbest à l'itération t et Gbets à l'itération t+1 est égale à une précision spécifiée, c'est à dire qu'il n'y a pas d'amélioration notable de la solution alors il y a arrêt des calculs.

Tous ces critères sont implémentés, paramétrables depuis l'interface utilisateur, et leurs valeurs varient selon le problème à optimiser.

Dans notre cas, nous avons lancé les tests en utilisant ces trois critères, en changeant à chaque fois leurs valeurs, afin de trouver les valeurs les plus optimales à notre problématique. Après avoir trouvé cette combinaison de valeurs, on lance l'exécution et une fois l'un des critères est atteint, le programme s'arrête.

3.4 Application de l'algorithme MPSO au problème de transport d'électricité

Dans le but de tester les performances de la méthode MPSO, nous l'avons appliqué au problème du pylône électrique. Le but de cette analyse est de déterminer les efforts, les contraintes dans les différents éléments du treillis. Afin de définir le déplacement maximal engendré par les charges appliquées et de vérifier si certains éléments du treillis sont sujets au flambement (voir Figure 3-1).

En effet, une conception et optimisation s'avère nécessaire pour trouver la section droite

optimale de chaque barre, dans le but de minimiser le déplacement du pylône et d'améliorer d'avantage sa fiabilité.

Les variables d'optimisation sont X= {H,W}, où H et W sont respectivement la hauteur et la largeur de la section droite de la barre, et qui sont les composantes principales de notre fonction objectif.

3.4.1 Modélisation du problème

L'objectif de ce probème est de concevoir des barres du treillis qui ont une structure en acier et un poids minimal (volume) pouvant supporter des charges appliquées sur le support dans les limites du déplacement admissible.

Les propriétés du matériau sont comme suit:

1. Module d'élasticité EX = 200 GPa,
2. Coefficient de Poisson PRXY = 0.3,
3. Masse volumique ρ de 7500 kg / m^3.

Les longueurs des membres du trellis sont fixes. Les variables de conception ou les variables d'entrée pour le problème d'optimisation sont les sections transversales des membres de la structure.

Les sections des éléments de la structure peuvent varier entre 5 et 100 mm^2. La limite de déplacement est de 9 mm dans toutes les directions.

La géométrie du problème est illustrée à la Figure 3-1. La structure est articulée au niveau des nœuds (1, 2).

L'élément ANSYS LINK1 est utilisé pour modéliser, simuler et analyser la structure du trellis Les membres de la structure sont divisés en quatre groupes d'éléments, comme illustré dans la Figure 3- 6. [56].

Deux charges identiques F de 1,8 KN sont appliquées aux deux extrémités supérieures du pylône (voir Figure 3-1) [57].

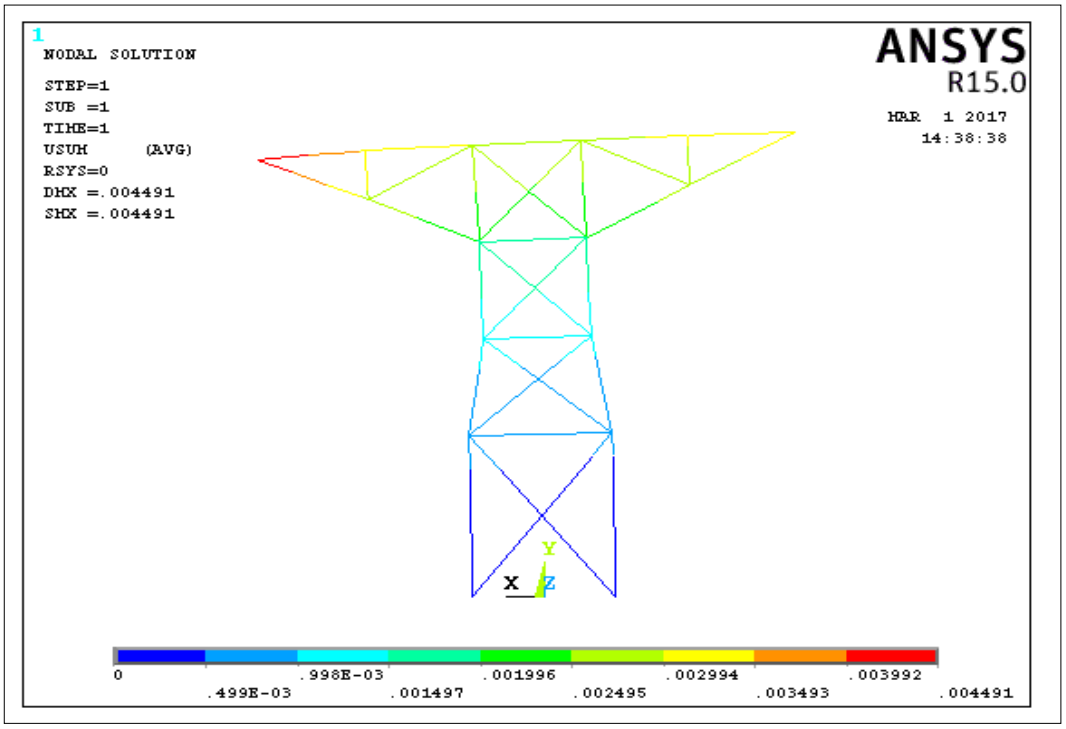


Figure 3-3 Analyse du déplacement maximal du pylône.

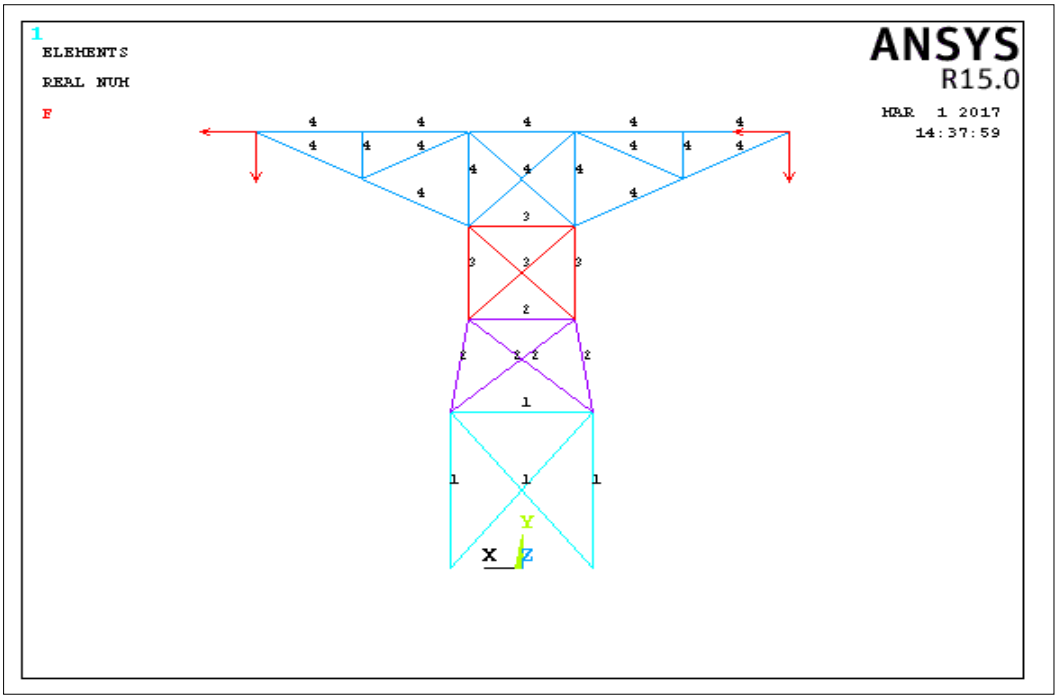


Figure 3-4 Représentation des quatre groupes d'éléments.

3.4.2 Formulation mathématique du problème

Les quatre groupes d'éléments désignent quatre variables de conception. L'espace de conception est en quatre dimensions où chaque variable de conception (surface en coupe transversale) est représentée par quatre coordonnées de l'espace de conception.

La formulation mathématique peut être exprimée comme suit:

$$A = [A_1, A_2, A_3, A_4]^T$$

$$f = \min W = \sum_{1}^{4} \rho_i \, . A_i \, . L_i$$

Avec: $g(A) = \text{ umax-uc} \leq 0$

$$A_{min} \leq A \leq A_{max}$$

Où :

- $A = [A_1, A_2, A_3, A_4]^T$ sont les variables de conception pour la section.
- f est la fonction objectif.
- W est le poids total de la structure.
- ρ_i , A_i et L_i représentent respectivement la densité, l'aire de la section, la longueur de la barre du groupe de barres.
- g(A) représente les contraintes de déplacement.
- umax et uc, représentent respectivement le déplacement maximal et la limite de déplacement du groupe de barres, dans différentes conditions.
- Amin et Amax correspondent respectivement aux tailles minimale et maximale des sections.

3.5 Résultats et discussion

Les résultats obtenus relatifs à la résolution de ce problème d'optimisation à l'aide de l'algorithme proposé MPSO sont ensuite comparés aux résultats de la technique d'optimisation conventionnelle de premier ordre du logiciel ANSYS.

L'optimisation de premier ordre ANSYS est une méthode d'optimisation conventionnelle dans laquelle les vraies fonctions (objectif et contraintes) sont utilisées pour l'optimisation.
Un fichier de paramétrage APDL (ANSYS Parametric Design Language) du logiciel ANSYS permet la définition des phases du pré-traitement, de la solution, du post-traitement et d'optimisation.
Dans la phase d'optimisation, l'algorithme d'optimisation MPSO est utilisé. Ensuite, le nombre d'itérations, les limites supérieures et inférieures des variables de conception et les limites des contraintes sont spécifiées. Le processus converge au niveau de l'itération spécifiée si une valeur minimale de la fonction objectif a été trouvée. Si elle ne converge pas, une seconde itération doit être configurée avec un nouveau point de départ. Cela se poursuit jusqu'à ce qu'une convergence soit rencontrée dans les contraintes spécifiées.

Le même problème a également été résolu à l'aide de la méthode d'optimisation de premier ordre d'ANSYS. Le nombre maximal d'itérations a été défini à 20.

Le problème d'optimisation a convergé vers une valeur minimale de la fonction objectif donnée par W = 1.46E7 Kg. Les variables de conception optimales sont :

- A_1 = 26,86 mm²,
- A_2 = 25,95 mm²,
- A_3 = 24,86 mm²,
- A_4 = 24,54 mm².

Le graphique de convergence pour l'optimisation ANSYS du premier ordre et les valeurs optimales des variables de conception sont présentés à la figure 3-7.

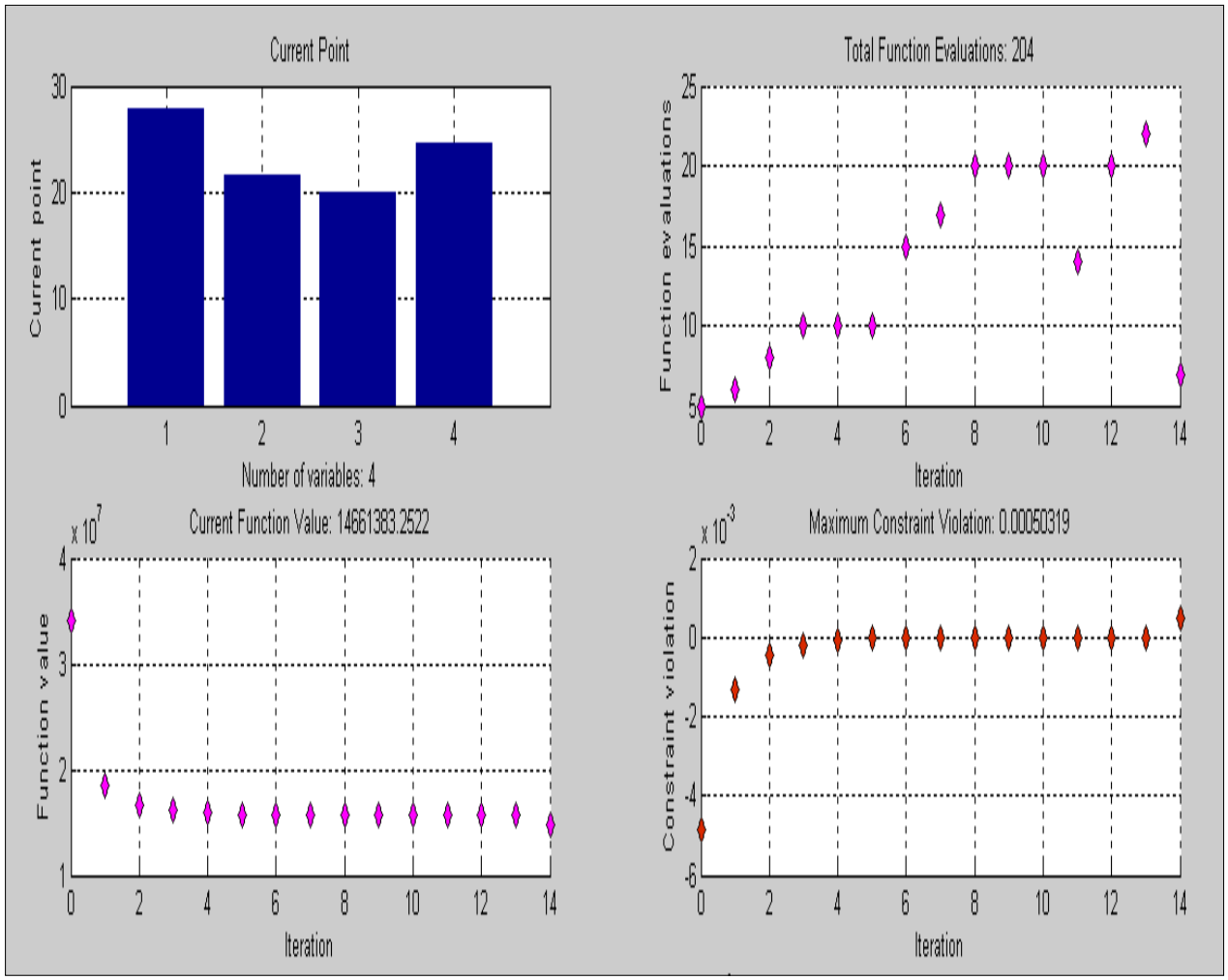


Figure 3-5 Diagramme de convergence des fonctions.

3.5.1 Étapes de l'application de MPSO du transport d'électricité

Pour l'algorithme MPSO proposé, la procédure d'optimisation peut être résumée dans les étapes suivantes :

- Lire le fichier de configuration APDL ANSYS.

- Créer une réponse en utilisant une analyse de régression basée sur un polynôme de second ordre en utilisant les modèles candidats et les réponses vraies. Le diagramme sous forme de surfaces de réponse générées pour la contrainte de déplacement est illustré à la Figure 3-6.

- Combinez l'algorithme MPSO proposé directement sur le digramme de réponse (surface) et évaluez la solution de conception optimale.

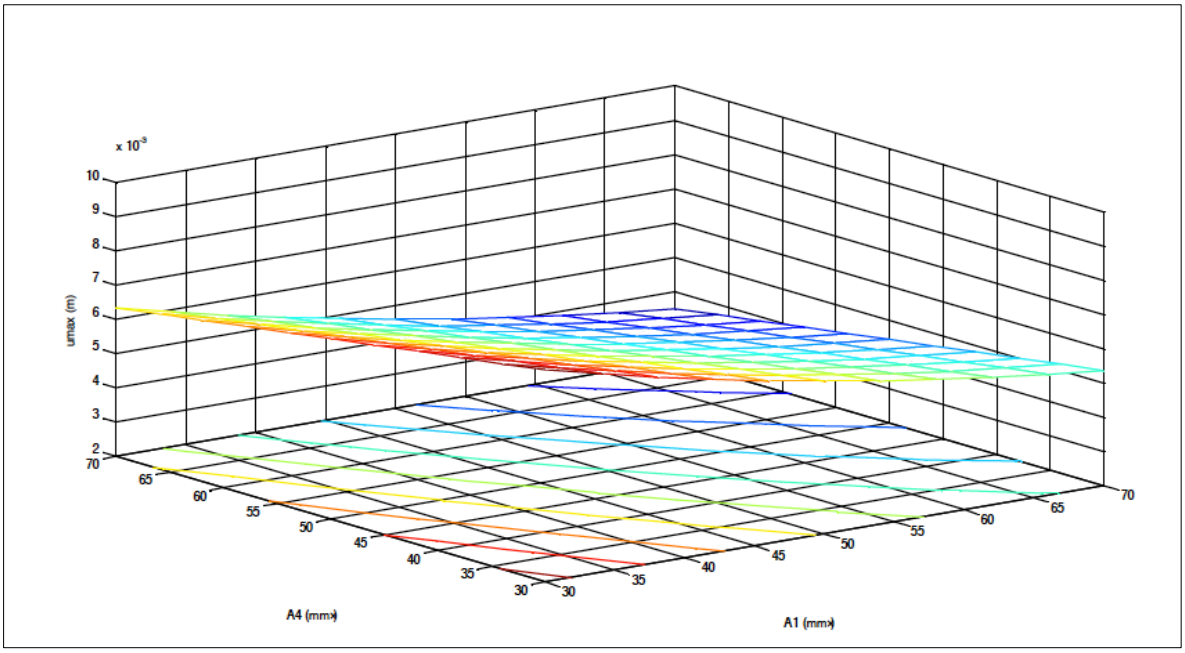


Figure 3-6 Diagramme de réponse pour le déplacement maximal.

Le modèle MPSO est doté d'une interface utilisateur permettant la configuration manuelle des différents paramètres MPSO utilisés selon le besoin de l'utilisateur. (Voir Figure 3-9)

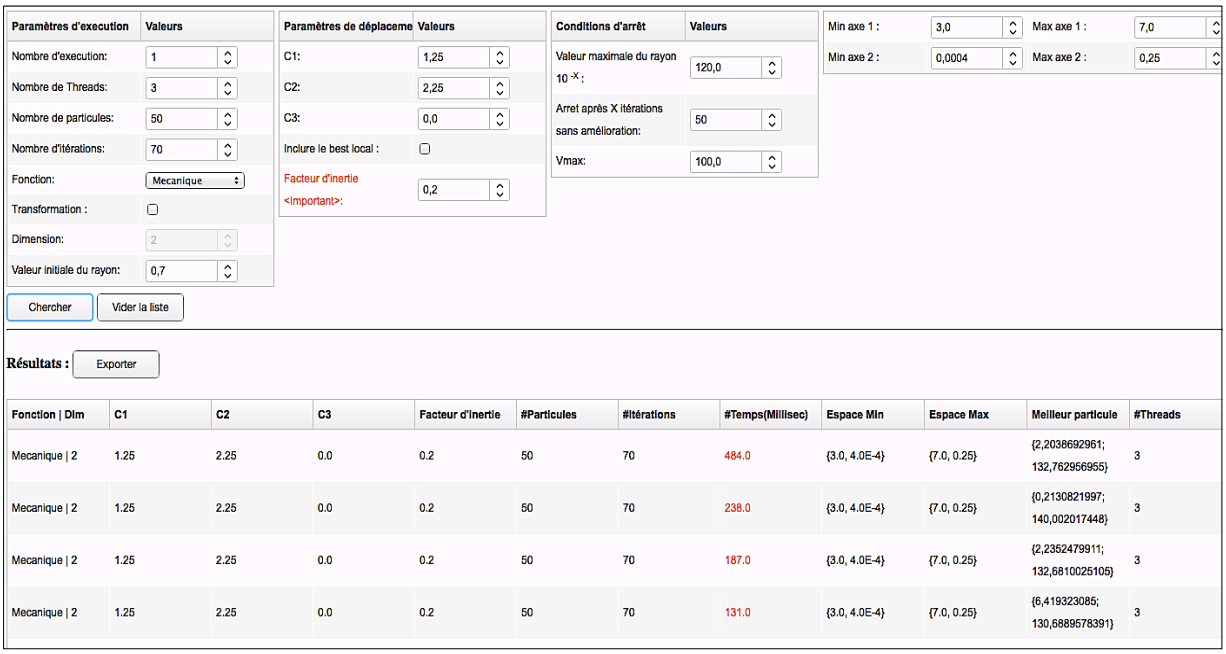


Figure 3-7 Capture d'écran de l'interface utilisateur pour le modèle MPSO.

3.5.2 Résultats numériques

Les résultats obtenus pour ce problème en utilisant l'algorithme MPSO sont présentés dans le tableau 3-1.

Tableau 3-1 Point de conception de l'algorithme MPSO.

Paramètres	Point initial	ANSYS	Algorithme MPSO
A1 (mm)	60	26.85	27.05
A2 (mm)	55	25.94	25.34
A3 (mm)	55	24.86	24.14
A4 (mm)	50	24.54	24.91
Umax (m)	0.0025	0 .0085	0 .0089
Fobj (Kg)	3.402E7	1.466E7	1.451e+007

On observe ici que les résultats pour la fonction objectif du processus d'optimisation MPSO sont meilleurs que ceux du processus d'optimisation de premier ordre ANSYS.

3.6 Conclusion

Ce chapitre contient le descriptif de l'implémentation d'une approche parallèle avec voisinage évolutif de la méta-heuristique PSO, appliquée au problème de transport de l'électricité.

Le modèle parallèle MPSO a été utilisé dans cette problématique dans le but d'améliorer la qualité de la solution, à travers la bonne exploration et exploitation de l'espace de recherche (grâce à la notion de voisinage évolutif) ceci consomme beaucoup de temps de traitement. Ce temps est réduit grâce au calcul parallèle.

L'algorithme MPSO proposé a été implémenté en se basant sur l'échange entre les sous-essaims. Les résultats obtenus, en résolvant ce problème de transport de l'électricité à l'aide de l'algorithme MPSO, sont comparés aux résultats de la technique d'optimisation classique du premier ordre du logiciel ANSYS.
À travers les résultats obtenus, nous constatons que notre modèle présente une efficacité remarquable en terme de qualité de la solution trouvée.

Une nouvelle variante de ce modèle est développée dans le chapitre suivant, basée sur l'hybridation de MPSO avec la méthode du recuit simulé SA, et qui sera appliquée au problème des réseaux de collaboration.

Chapitre 4Élaboration d'un modèle d'optimisation hybride appliqué au problème des réseaux de collaboration

Sommaire

4.1 Introduction

Dans le but d'instaurer ou d'améliorer les réseaux de collaboration dans les organismes, il est primordial d'effectuer une analyse et un diagnostic de la situation actuelle afin d'en conclure les obstacles et les pistes d'amélioration.

Plusieurs études ont été menées dans ce sens. Un ensemble d'approches d'évaluation et de métriques ont été proposées ; nous distinguons deux types : les approches qui se basent sur des évaluations qualitatives (mesures de maturité) et d'autres quantitatives (mesures de compatibilité).

Dans ce chapitre, H-MPSO-SA une nouvelle variante du modèle MPSO est proposée. Elle est basée sur l'hybridation de MPSO avec la méta-heuristique SA dans le but améliorer d'avantage la qualité des résultats obtenus pour le modèle MPSO.

Le modèle hybride H-MPSO-SA a été testé en premier lieu sur un ensemble de fonctions tests, et suivant les résultats obtenus une amélioration de la qualité de la solution obtenue a été remarquée. Ensuite H-MPSO-SA a été appliqué à la problématique des réseaux de collaboration. Le détail du dit modèle ainsi que son application seront développés dans ce chapitre.

4.2 Contexte général des réseaux de collaboration des SI

4.2.1 Concept de l'interopérabilité

L'interopérabilité est utilisée dans la majorité des domaines. Plusieurs définitions lui ont été attribuées, nous proposons : « la capacité de deux ou plusieurs systèmes ou éléments à échanger des informations et d'utiliser l'information qui a été échangée » [58]. Ce qui signifie que le but de l'interopérabilité est de créer des services adéquats permettant aux systèmes hétérogènes de s'échanger correctement.

Pour notre cas, nous allons nous intéresser à une notion de mesure du degré d'interopérabilité en utilisant une approche se basant sur 3 principaux axes :

1. Le niveau de maturité de l'interopérabilité relatif à l'environnement contenant les systèmes d'information à analyser ;
2. Le degré de compatibilité des interfaces externes relatifs aux systèmes d'information entre eux ;
3. La performance opérationnelle de l'interopérabilité en se focalisant sur les trois aspects suivants : qualité, coût et délai.

4.2.2 Classification de l'interopérabilité

Vu que l'interopérabilité est primordiale dans différents secteurs, à savoir : l'industrie en général, l'informatique, le médical, l'aérospatiale, le réseau ferroviaire, etc. Plusieurs classifications de cette notion on été proposées. Dans ce manuscrit une classification illustrative est représentée dans la Figure 4-1 :

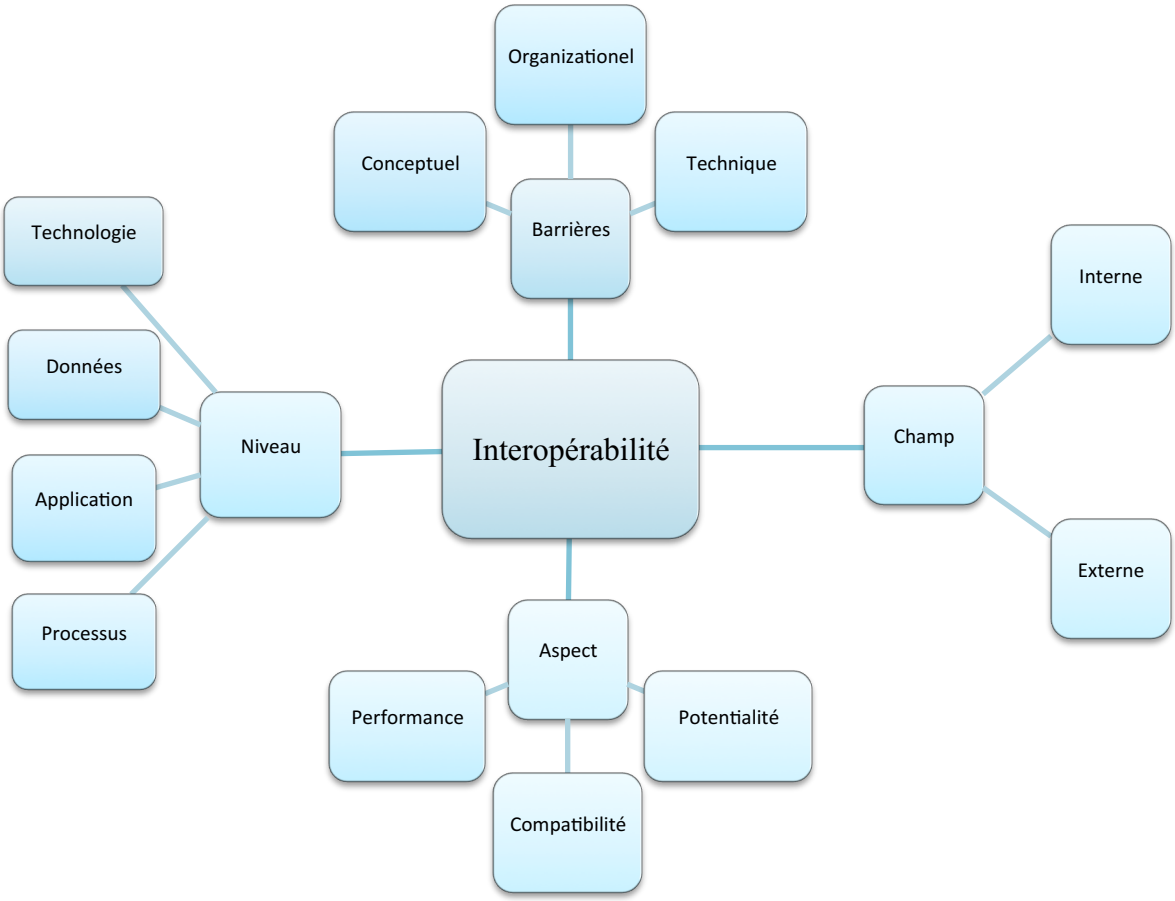


Figure 4-1 Classification de l'interopérabilité.

4.2.3 Métrique de l'interopérabilité

Comme mentionné précédemment, le modèle proposé pour la détermination du degré de l'interopérabilité est basé sur trois grands axes cités ci-dessus tout en prenant en considération les différents niveaux de l'interopérabilité dans le but d'en éradiquer l'ensemble des obstacles qui empêchent le bon fonctionnement.

Quatre étapes sont à distinguer dans l'approche proposée :

1. Quantifier le potentiel d'interopérabilité ;
2. Calculer la compatibilité de l'interopérabilité ;
3. Déterminer la performance de l'interopérabilité ;

4. Définir le ratio global de l'interopérabilité.

4.2.3.1 Potentiel d'interopérabilité

Dans le but de calculer le potentiel d'interopérabilité P, une mesure proposée dans [59] est utilisée, en proposant le mappage décrit dans le tableau ci-dessous. (Voir Tableau 4-1).

Tableau 4-1 Quantification du potentiel d'interopérabilité.

Niveau de maturité (NM)	Quantification de la potentialité
1	0.2
2	0.4
3	0.6
4	0.8
5	1

On utilise la formule 4-1 pour calculer le potentiel d'interopérabilité :

$$P = 0.2 * NM \qquad (4\text{-}1)$$

4.2.3.2 Compatibilité de l'interopérabilité

Dans le but de calculer la compatibilité de l'interopérabilité CI, on utilise une version modifiée de la matrice proposée dans [60] (voir Tableau 4-2) ; cette dernière est composée d'une fusion des « niveaux » et des « barrières » de l'interopérabilité.

Tableau 4-2 Calcul du degré de compatibilité de l'interopérabilité.

	Conceptuel		Organizationel		Technologie	
	syntaxique	**sémantique**	**Résponsabilités des autorités**	**Organization**	**Plate-forme**	**Communication**
Commerce	m_{11}	m_{12}	m_{13}	m_{14}	m_{15}	m_{16}
Processus	m_{21}	m_{22}	m_{23}	m_{24}	m_{25}	m_{26}
Service	m_{31}	m_{32}	m_{33}	m_{34}	m_{35}	m_{36}

Données	m_{41}	m_{42}	m_{43}	m_{44}	m_{45}	m_{46}

Si dans une zone donnée les critères marquent satisfaction, alors la valeur 1 est affectée, sinon la valeur 0 est affectée dans le cas contraire (zone contenant une multitude d'incompatibilités).

Notons le degré élémentaire de la compatibilité M, i prend des valeurs de 1..4, et j prend des valeurs de 1..6. Le degré de compatibilité M est calculé en utilisant la formule 4-2:

$$M = \sum_{i}\sum_{j} {}^{m_{ij}}\!/_{24} \qquad (4\text{-}2)$$

4.2.3.3 Détermination de la performance

Pour évaluer la performance opérationnelle P', on calcule pour chaque type le degré de sa performance opérationnelle en prenant en considération : la qualité de l'ensemble des informations communiquées, les charges dues à la modification des systèmes dans le but d'obtenir une bonne qualité d'interopérabilité et un meilleur temps, et enfin la durée d'interopérabilité.

- **Durée d'interopérabilité**

La durée d'interopérabilité est relative à la durée entre la date de la demande d'information et la date à laquelle l'information a été utilisée.

On peut distinguer différentes périodes de temps [61]:

- « Request time » cette période correspond à la durée étalée entre la date d'envoi de la demande et la date de la réception par le destinataire.
- « Treatment time » représente le temps pris pour le traitement de la demande.
- « Return time » c'est la période relative à la durée entre la date du renvoi de la demande d'information et la date de la réception de l'information renvoyée.

- « Time to use » c'est la durée prise entre la réception de l'information et l'utilisation (exploitation) de cette dernière.

Pour chaque période, on calcule sa valeur en comparant le temps fourni pas SLA (Service-Level Agreement) et le temps réel en utilisant la formule 4-3 :

$$t_{réel} > t_{SLA} \Rightarrow t = t_{SLA} / t_{réel} \qquad (4\text{-}3)$$

Si le temps calculé est plus long que le temps prévu, alors il y a une carence, sinon s'il est respecté, alors on accorde 1 à la valeur du temps (voir formule 4-4) :

$$t_{réel} \leq t_{SLA} \Rightarrow t = 1 \qquad (4\text{-}4)$$

Pour chaque période, on calcule sa valeur en comparant le temps fourni pas SLA (Service-Level Agreement) et le temps réel en utilisant la formule 4-5 :

$$t_{réel} > t_{SLA} \Rightarrow t = t_{SLA} / t_{réel} \qquad (4\text{-}5)$$

Si le temps calculé est plus long que le temps prévu, alors il y a une carence, sinon s'il est respecté, alors on accorde 1 à la valeur du temps (voir formule 4-6) :

$$t_{réel} \leq t_{SLA} \Rightarrow t = 1 \qquad (4\text{-}6)$$

La définition de la valeur de la durée de l'interopérabilité est calculée en utilisant la moyenne géométrique de toutes les périodes de temps qui composent ce dernier (voir la formule 4-7).

$$TI = \sqrt[4]{(t_{req} \times t_{treat} \times t_{ret} \times t_{use})} \qquad (4\text{-}7)$$

- **Coût de l'interopérabilité**

La définition du coût de l'interopérabilité est calculée en utilisant le coût de l'échange C_e et le coût nécessaire pour rendre les informations échangées utilisables C_u. Pour l'évaluer on

compare sa valeur réelle avec celle fournie par SLA déclarée (coût de référence). (Voir la formule 4-8).

$$CI = \sqrt{(C_{ex} \times C_{use})} \qquad (4\text{-}8)$$

- **Qualité de l'interopérabilité**

Il existe 3 principaux types à prendre en considération pour la qualité de l'interopérabilité, nous distinguons : la qualité d'échange, de conformité et d'utilisation.

1) La qualité d'échange comme son nom l'indique est relative à l'échange d'informations, elle est établie si l'échange est correctement effectué (voir formule 4-9):

$$q_{ex} = n_{succ} / n_{tot} \qquad (4\text{-}8)$$

2) La qualité de conformité est relative au nombre d'informations conformes, comparées au nombre des informations reçues (voir formule 4-10):

$$q_{conf} = n_{conf} / n_{tot} \qquad (4\text{-}9)$$

3) La qualité d'utilisation représente l'exploitation de l'information, une vérification de l'utilisation de l'information reçue (voir formule 4-11) :

$$q_{use} = n_{use} / n_{rec} \qquad (4\text{-}10)$$

4.2.4 Ratio global de l'interopérabilité

En prenant en considération la nature des trois indicateurs précédemment cités, l'utilisation de la moyenne arithmétique comme fonction d'agrégation pour calculer le degré global de l'interopérabilité (voir formule 4-12).

$$Ratio = \frac{(PI + M + PO)}{3} \qquad (4\text{-}11)$$

Si le département informatique est doté des éléments pour pondérer l'ensemble de ces trois indicateurs avec des poids différents (w_1, w_2, w_3) ; on opte alors pour la moyenne arithmétique pondérée. (Voir formule 4-13)

$$Ratio = \frac{(w_1 * PI + w_2 * M + w_3 * PO)}{(w_1 + w_2 + w_3)} \qquad (4\text{-}12)$$

4.2.5 Description de la problématique

Optimiser les efforts de mise en œuvre des réseaux de collaboration est une condition essentielle pour établir, développer et faire évoluer efficacement la collaboration intra et inter organisationnelle. Ainsi, pour assurer cet objectif d'efficacité, nous proposons initialement une modélisation de l'évolution des réseaux de collaboration des systèmes d'information impliqués. Le degré d'interopérabilité, développé dans la section suivante, est évalué à l'aide d'un nouveau modèle d'optimisation H-MPSO-SA [62].

Le choix du modèle hybride H-MPSO-SA est du à la complexité de la problématique des réseaux de collaboration; ce qui impose l'utilisation d'un algorithme robuste permettant d'effectuer les calculs en parallèle pour éviter un long temps d'attente tout en garantissant une meilleure qualité de la solution. En plus, le calcul parallèle s'impose lors de l'utilisation des tableaux de bord de l'évolution des indicateurs de qualité (ratio d'interopérabilité) Les architectes de Systèmes d'Information ont besoin d'informations visuelles sur la matrice d'effort optimal en temps réel pour avoir une vision rapide afin de prendre les décisions.

4.3 Algorithme hybride proposé

Dans le but de d'améliorer la qualité des résultats obtenus du modèle MPSO, nous avons conçu une nouvelle variante hybride, nommée H-MPSO-SA.

4.3.1 Description de l'algorithme H-MPSO-SA

H-MPSO-SA est basée sur l'idée que l'algorithme MPSO assure une convergence rapide, tandis que l'algorithme SA permet d'échapper aux optima locaux dus à sa forte capacité de recherche locale. SA n'est pas utilisé à chaque itération de notre algorithme H-MPSO-SA, mais il exécute son traitement après un nombre spécifique d'itérations (prédéfini avant le démarrage du programme), puis l'algorithme PSO poursuit ses calculs. Nous avons constaté que c'est plus judicieux que de l'utiliser à chaque itération.

Si, à chaque itération, nous avons utilisé à la fois PSO et SA, SA tentera de diversifier les points et en même temps PSO tentera de faire converger les points, ce qui retardera à son tour la convergence de PSO ainsi que les capacités de SA, qui également ne sera pas efficace.

Pour cela, dans H-MPSO-SA, nous commençons par appliquer MPSO, et ensuite appliquer SA après chaque «k» PSO itérations à la meilleure solution de l'essaim, afin de permettre à SA de sortir des optima locaux et de diversifier les particules prématurément convergées dans l'espace de recherche.

Le modèle hybride proposé est conçu pour des problèmes d'optimisation complexes, avec un large espace de recherche et différents optima locaux. Il est capable de maintenir une convergence rapide (la plupart du temps) grâce au MPSO parallèle et de sortir d'un optimum local à l'aide de l'algorithme SA.

L'hybridation de ces deux algorithmes selon notre modèle proposé H-MPSO-SA a réussi à améliorer les performances de l'algorithme MPSO en terme de la qualité de la solution et a mené à des résultats satisfaisants.

L'organigramme du modèle hybride H-MPSO-SA est illustré dans la Figure 4-2 [62].

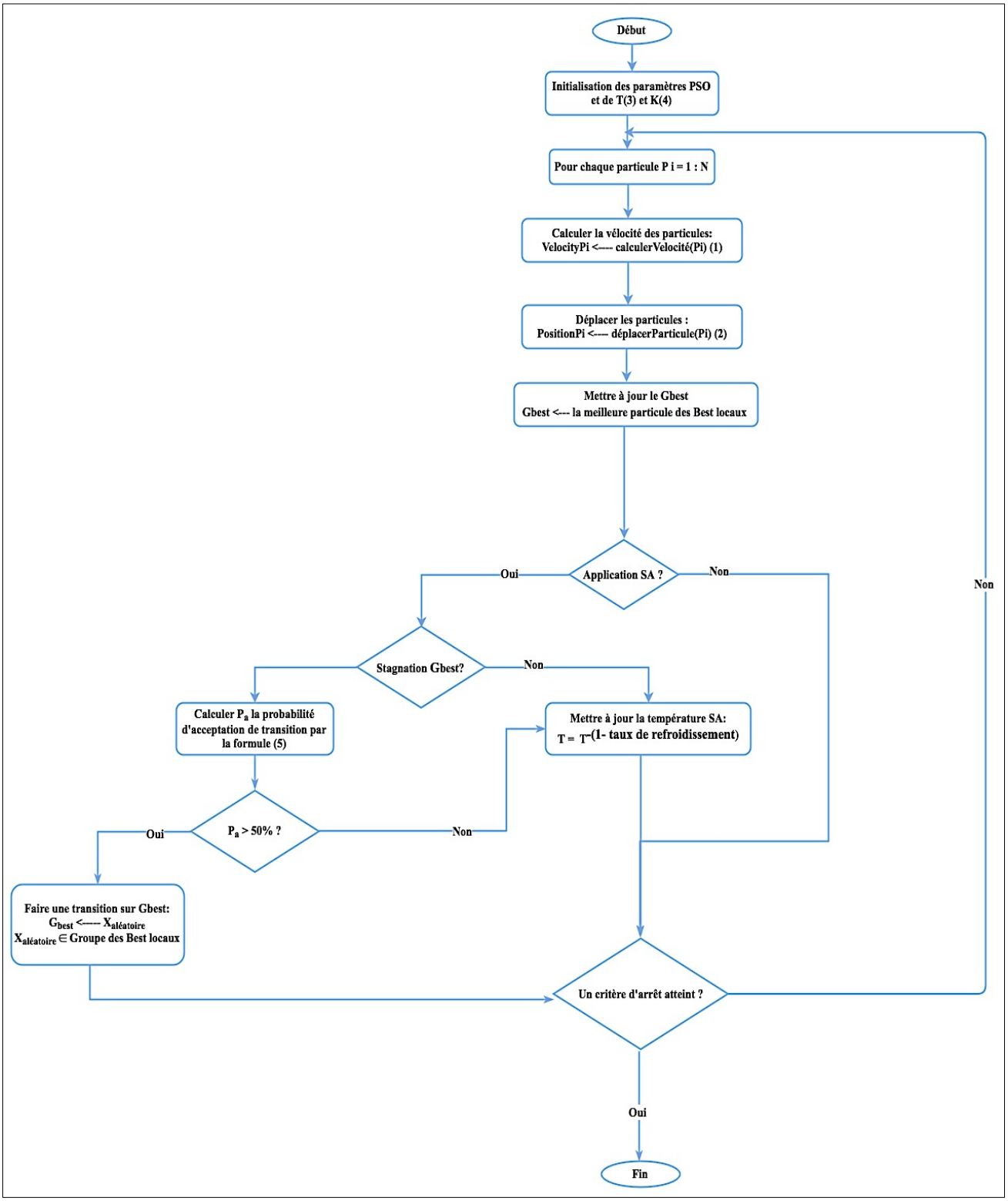


Figure 4-2 Organigramme du modèle hybride H-MPSO-SA.

Avec:

(1): $V_{iD(t+1)} = V_{iD(t)} + C_1 r_1 (Pb_{iD(t)} - X_{iD(t)}) + C_2 r_2 (Pg_{iD(t)} - X_{iD(t)})$

(2): $X_{iD(t+1)} = X_{iD(t)} + V_{iD(t)}$

(3): «T» la température initiale du système (elle diminue après chaque itération pour stabiliser le système).

(4): «k» la fréquence d'application de l'algorithme SA afin d'optimiser le temps de calcul.

(5): La probabilité d'acceptation = Exp ((différence-énergie) / (Température))

$$= e^{-\Delta E/T}$$

Avec:

Exp: la fonction exponentielle.

Différence-énergie = énergie du meilleur global - énergie du meilleur aléatoire local.

Température: température actuelle de l'algorithme SA.

4.3.2 Expérimentations et résultats de H-MPSO-SA sur les fonctions tests

Pour tester les performances du modèle hybride proposé, nous l'avons testé tout d'abord sur un ensemble de fonctions test (voir Tableau 4-3) avant de le tester sur un problème d'optimisation réel.

Les fonctions utilisées ont été appliquées à différents processus d'optimisation et constituent une source fiable pour mesurer la qualité des algorithmes d'optimisation. Pour chacune de ces fonctions, il existe de nombreux optima locaux ainsi qu'un ou plusieurs optima globaux dans leur espace de recherche. Au fur et à mesure que nous augmentons le nombre de dimensions, le problème devient plus complexe, plus d'optima locaux risquent de se produire et conduisent à un retard dans la convergence vers la solution globale. Pour la présente étude, 2, 10 et 30 dimensions ont été utilisées.

Tableau 4-3 Descriptif des fonctions utilisées dans nos expérimentations.

Fonction	Domaine	f_{min}	Dimension
f_1 Sphere	±5,12	0	30
f_2 Griewank	±600	0	30
f_3 Rosenbrock	±30	0	30
f_4 Rastring	±5.12	0	30
f_5 Schwefel	±500	0	30
f_6 Ackley	±32	0	30
f_7 Michalewicz	±π	-9.66015	10
f_8 Shubert	±10	-186.739	10
f_9 Step	±100	0	30
f_{10} Himmelblau	±30	-3.78396	2

Comme mentionné précédemment, notre programme dispose d'une IHM, qui permet à l'utilisateur de choisir et de modifier les paramètres PSO et SA selon ses besoins, à savoir : le problème à optimiser (la fonction objectif), le nombre de particules, le nombre des itérations, la dimension du problème, la topologie de communication, la valeur initiale du rayon, la valeur initiale de la température, le taux de refroidissement, le critère d'arrêt (voir Figure 4-3).

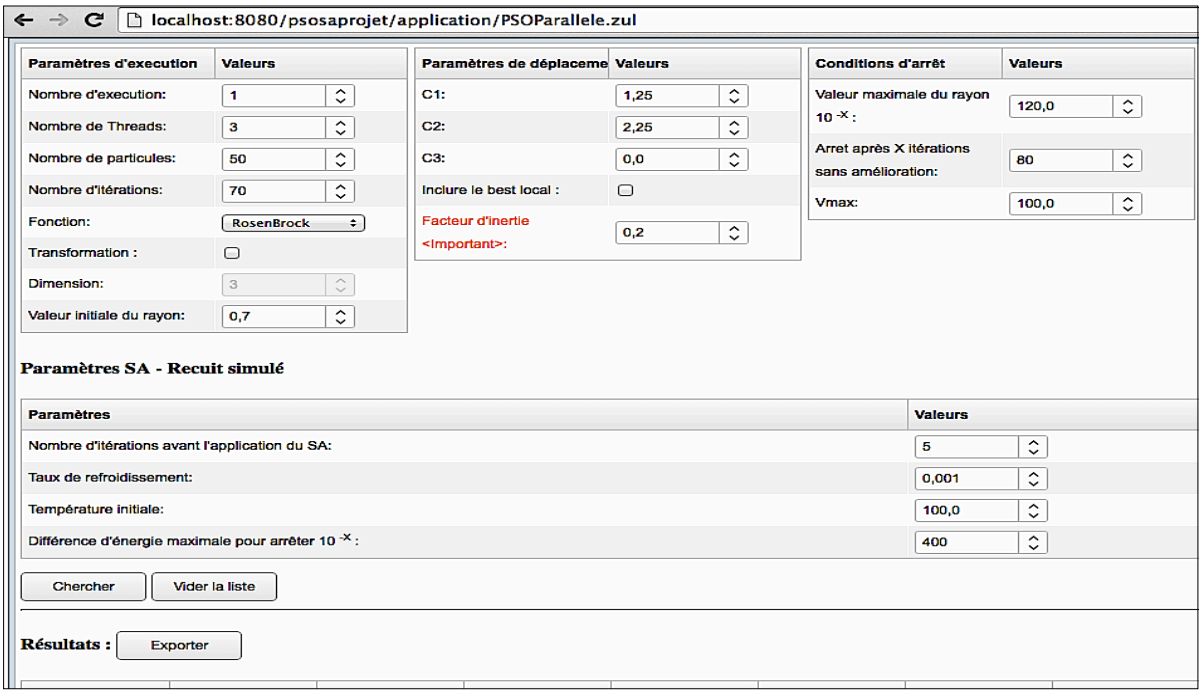


Figure 4-3 Capture d'écran de l'interface utilisateur pour le modèle H-MPSO-SA

Ces paramètres changent en fonction de la complexité du problème à optimiser, le tableau ci-dessous contient la liste des paramètres utilisés dans nos expérimentations et qui ont donné des résultats satisfaisants.

Pour le nombre de threads utilisés, il dépend du nombre de voisinages créés à chaque itération, puisque chaque thread s'occupe du traitement d'un voisinage, et le nombre de voisinages change à chaque itération selon des critères relatifs au nombre existant de voisinages.

La valeur du rayon dépend aussi du problème d'optimisation, plus précisément l'espace de recherche de la fonction à optimiser, mais son choix reste critique, puisqu'une petite valeur de rayon est égale à un grand nombre de voisinages, alors qu'une grande valeur permet d'obtenir un nombre limité de voisinages.

Pour les paramètres SA, la valeur de la température initiale est très importante, dépend aussi du problème à optimiser, et lié au taux de refroidissement, le choix de ces deux valeurs doit respecter cette liaison.

La valeur de « k » est aussi importante, dépend des contraintes du problème d'optimisation, son choix est effectué en prenant en considération les paramètres PSO configurés, spécialement le nombre des itérations.

Tableau 4-4 Descriptif des paramètres PSO et SA utilisés dans nos expérimentations.

Paramètres PSO & SA	Modèle H-MPSO-SA
Nombre de particules	30
Nombre des itérations	1500
Coefficients d'accélération	C1 = 1.25, C2=2.25, C3=2.25
Facteur d'inertie	(0.4 – 0.2)
Topologie de communication	Anneau
Nombre de threads	Dépend des voisinages crées
Temparature initiale	100
Taux de refroidissement	0.001
Valeur de "k"	5
Valeur du rayon	Dépend de la fonction objectif

Les expérimentations effectuées sur un lot de 10 fonctions tests. En moyenne, nous avons utilisé 100 000 évaluations de fonctions.

Selon les résultats obtenus, on peut dire que l'algorithme H-MPSO-SA fournit la solution optimale avec une probabilité plus élevée et que le temps de calcul dans H-MPSO-SA est inférieur à celui du PSO et du MPSO de base. Les résultats graphiques sont illustrés dans les figures ci-dessous.

Figure 4-4 Courbes de performance du temps de calcul du PSO, MPSO et H-MPSO-SA.

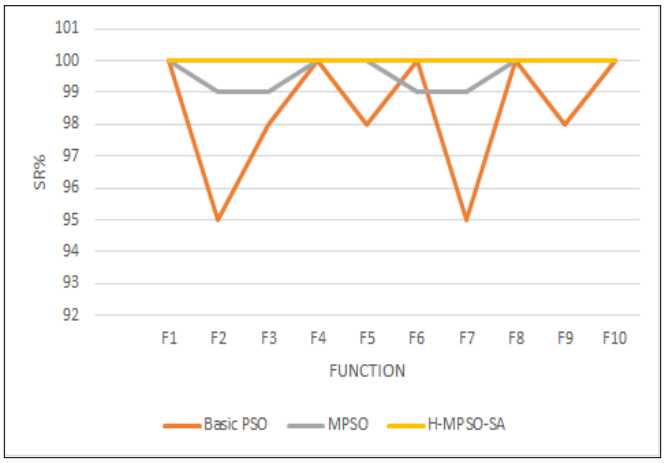


Figure 4-5 Courbes de performance du “SR” taux de succès du PSO, MPSO et H-MPSO-SA.

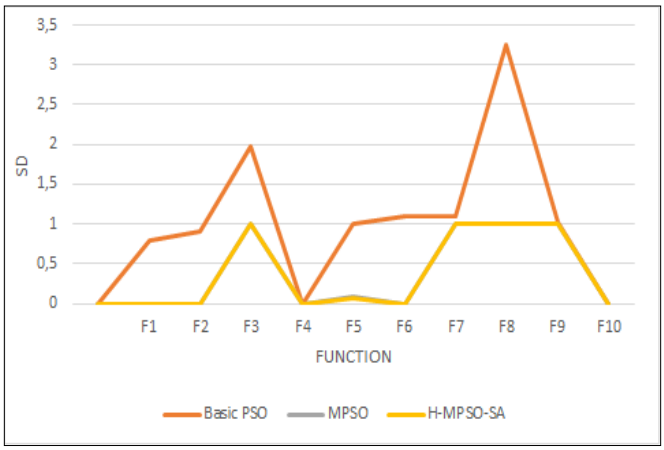


Figure 4-6 Courbes de performance du “SD”
l’écart type PSO, MPSO et H-MPSO-SA.

4.4 Proposition d’un modèle d’optimisation dans les réseaux de collaboration

4.4.1 Modélisation dans les réseaux de collaboration

La modélisation vise dans un premier temps à quantifier le degré d'interopérabilité d'un système d'information au sein de son écosystème de manière scalaire. Ensuite une amélioration de l'interopérabilité d'un système d'information est obtenue par le couplage de l’approche proposée et de la modélisation linéaire. Le résultat obtenu incite à une évolution du système général d’interopérabilité dans un ensemble de systèmes d’information interconnectés permettant une surveillance des efforts nécessaires pour améliorer le degré d’interopérabilité dans les réseaux de collaboration.

L’ensemble des «n» systèmes (S_1, S_2,…, S_n) ci-dessous doit être pris en compte:

- S_i ($_i = {}_1 \ldots {}_n$) l’ensemble des systèmes d’information.
- Les n systèmes d’information communiquent entre eux.

a_i = Ratio qui représente le niveau d'interopérabilité du système S_i.

I = (a_i) est le vecteur actuel d'interopérabilité

I’= (a'_i) est le vecteur cible de l’interopérabilité (voir la formule 4-14).

$$a'_i = \sum E_{ij}\, a_j \qquad (4\text{-}13)$$

E_{ij} représente l'effort à appliquer sur le système Si pour améliorer le système S_j.

$E = (E_{ij})$ est la matrice d'effort à appliquer pour atteindre le degré d'interopérabilité cible.

$I' = E\, I$.

Où:

$$\begin{pmatrix} E_{11} & E_{12} & \dots & E_{1n} \\ E_{21} & E_{22} & \dots & E_{2n} \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots \\ E_{n1} & E_{n2} & \dots & E_{nn} \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ \dots \\ \dots \\ a_n \end{pmatrix} = \begin{pmatrix} a'_1 \\ a'_2 \\ \dots \\ \dots \\ a'_n \end{pmatrix}$$

Si tous les systèmes S_i sont compatibles les uns avec les autres et s'il n'y a pas de barrière explicite empêchant l'interaction entre les systèmes, E_{ij} est équivalent au ratio de la charge de travail N_{ij}, qui correspond à la charge de travail allouée aux interfaces externes à améliorer S_i par rapport S_j.

Noverall est la charge de travail globale a été allouée pour améliorer l'interopérabilité.

$E_{ij} = N_{ij} / \text{Noverall}$

Dans ce cas, l'objectif ultime est d'atteindre le vecteur d'interopérabilité optimal.

Par conséquent, la formule (4-15) a été utilisée:

$$a'_i - \sum_i \sum_j E_{ij} \cdot a_i \le 0 \qquad (4\text{-}14)$$

Les contraintes sont pour chaque j:

$$\sum_i E_{ij} \le 100\%$$

Eij doit être multiplié par Ni / Noverall avec $Ni = \sum Nij$

4.4.2 Application de H-MPSO-SA au problème des réseaux de collaboration

À ce stade, on obtient la répartition optimale des efforts afin d'établir une situation de collaboration organisationnelle spécifique. La réconciliation du domaine d'amélioration de l'interopérabilité avec la répartition efficace des efforts dans un cadre multi-projets entre systèmes d'information s'appuyait sur un modèle linéaire innovant permettant le suivi et la planification de l'interopérabilité. Pour améliorer d'avantage les résultats obtenus, on utilise l'algorithme H-MPOS-SA sur la fonction objectif (voir la formule 4-15). Les résultats sont fournis par une matrice optimisée permettant une bonne estimation de l'effort requis pour l'interfaçage et l'interconnexion des systèmes d'information impliqués.

4.4.2.1 Description du cas pratique

Pour bien illustrer l'application de la méthode proposée, prenons le cas de quatre processus automatisés qui interagissent au sein d'une organisation (voir la Figure 4-7) [63]. Ces processus sont:

- Ressources humaines (S1);
- Comptabilité (S2);
- Chaîne d'approvisionnement (S3);
- Audit et contrôle de gestion (S4)

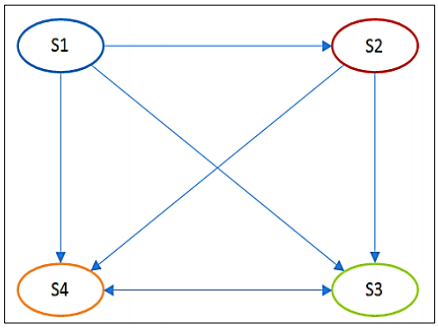


Figure 4-7 Modèle d'interaction entre les processus métier.

4.4.2.2 Formulation mathématique et résultats

On a proposé le degré d'interopérabilité de chaque système par les valeurs suivantes: 0,6, 0,45, 0,52 et 0,37, respectivement.

Les acteurs du système d'information, en accord avec les équipes métiers, ont pour objectif d'améliorer le degré d'interopérabilité de la collaboration au cours du trimestre suivant. Ils définissent également l'objectif à atteindre: 0,7, 0,6, 0,6 et 0,55, respectivement.

Pour atteindre cet objectif, nous utilisons l'algorithme H-MPSO-SA pour obtenir la matrice d'effort optimale qui réduit la fonction cible et obéit aux contraintes des équations (4-15) et (1-6). Par conséquent, nous avons obtenu la matrice suivante :

$$\begin{pmatrix} 88\% & 10\% & 17\% & 0\% \\ 81\% & 15\% & 3\% & 6\% \\ 66\% & 8\% & 30\% & 3\% \\ 44.5\% & 13.5\% & 27\% & 15\% \end{pmatrix}$$

4.4.2.3 Discussion des résultats

Le résultat de l'optimisation de la matrice d'effort permet aux architectes de systèmes d'information d'avoir une visibilité sur la solution optimale possible dans le domaine des solutions. La matrice de la méthode de distribution est optimisée avec succès, nous avons donc amélioré le degré d'interopérabilité.

Pour améliorer les résultats, en modifiant le vecteur cible, une nouvelle solution (matrice d'effort) peut être générée automatiquement. Le système d'optimisation est parfaitement configurable pour tout changement. Les résultats fournis sont un moniteur pour les architectes d'intégration, afin d'estimer l'efficacité des efforts nécessaires pour l'interopérabilité des systèmes d'information.

Pour avoir une bonne gouvernance des systèmes d'information, il est nécessaire de disposer d'une solution future convergeant vers une solution théorique. Si les architectes suggèrent une solution proche de notre résultat théorique, cette solution peut être acceptée. Sinon, le chef de projet invite les architectes à donner une autre estimation.

Le résultat présenté dans ce chapitre offre une connaissance de l'optimum des solutions possibles dans l'espace de recherche. Il aide les chercheurs à trouver une valeur acceptable et pratique à partir d'une solution mathématique.

Nous pouvons continuellement améliorer les résultats en ajustant le vecteur d'interopérabilité cible; et il appartient au système de générer une matrice de résultats différente.

Dans le présent document, le cas pratique utilise quatre processus. Lorsque nous dépassons ce nombre, nous ne pouvons pas résoudre le système de manière manuelle.

4.5 Conclusion

Ce chapitre étudie les réseaux de collaboration en tant que caractéristique de la qualité des interactions entre les systèmes d'information au sein des organisations.

Une proposition d'une approche pour modéliser et optimiser l'amélioration de l'interopérabilité des systèmes d'information consiste à planifier la répartition des efforts nécessaires afin d'établir une bonne collaboration. La réconciliation du domaine d'amélioration de l'interopérabilité avec la répartition efficace des efforts dans un cadre multi-projets entre les systèmes d'information s'appuyant sur un modèle linéaire innovant permettant le suivi et la planification de l'interopérabilité.

Dans le but d'améliorer les résultats obtenus de ce modèle, une variante de l'algorithme MPSO, basée sur l'hybridation de l'algorithme MPSO avec la méthode du recuit simulé SA a été utilisée. Pour évaluer les performances de l'algorithme proposé H-MPSO-SA a été testé sur un ensemble de fonctions tests, ensuite il a été testé pour améliorer la mise en œuvre des réseaux de collaboration. Les résultats obtenus ont été satisfaisants.

Les résultats obtenus sont fournis par une matrice optimisée qui permet une bonne estimation de l'effort requis pour l'interfaçage et l'interconnexion des systèmes d'information impliqués.

Conclusion et perspectives

Conclusion

L'émergence accélérée des problèmes complexes et de grande taille dans le monde réel a soulevé la demande des techniques de calcul parallèle. Cela a encouragé la recherche sur des algorithmes d'optimisation heuristiques tels que PSO, un algorithme basé sur l'intelligence en essaim, particulièrement adapté aux problèmes à variables continues, qui est largement appliqué à des problèmes du monde réel. La mise en œuvre des algorithmes PSO parallèles avec de nombreux modèles de parallélisation et de stratégies de résolution d'applications complexes a suscité une attention particulière de la part des chercheurs.

La méthode d'optimisation par essaim de particules PSO (Particle Swarm Optimization) proposé par J.Kennedy et R.Eberhart en 1995 a rapidement pris place parmi les méthodes les plus performantes dans la famille des méta-heuristiques dédiée à la résolution des problèmes d'optimisation complexes. Quoique cette méta-heuristique est très populaire par sa robustesse, mais cette dernière a des inconvénients dont les plus étudiés : le temps de calcul élevé et la convergence prématurée.

Dans cette thèse nous avons considéré diverses variantes de la méthode PSO qui visent à échapper aux deux inconvénients de la méthode. Les variantes proposées se basent sur deux concepts : la parallélisation des calculs et le jeux de voisinages scrutés sous divers angles. Une expérimentation des modèles proposés sur une série de fonctions tests a été effectuée pour évaluer les performances des algorithmes proposés. Les résultats expérimentaux obtenus été satisfaisants en termes de temps de calcul et de qualité de la solution.

Deux variantes parallèles MPSO et H-MPSO-SA ont été appliquées dans deux domaines différents : la transmission d'électricité et les réseaux de collaboration. Les résultats obtenus pour la résolution de ces deux problématiques été satisfaisants.

L'ensemble des contributions de cette thèse est structuré en quatre chapitres :
Le **premier chapitre** était consacré à l'établissement d'un état de l'art sur le concept du parallélisme en informatique en développant ses différents types, modèles et stratégies. La 2ème partie concernait la notion de l'optimisation et ses différents types. Les méthodes de résolution méta-heuristiques ont fait l'objet de la troisième partie du chapitre en développant les deux méta-heuristiques : POS et SA et qui sont utilisées dans nos expérimentations. Cette première phase de recherche, nous a permis d'acquérir une bonne base d'informations et de connaissances dans notre domaine de recherche. Ceci nous a orienté vers plusieurs axes de recherche et nous a ainsi permis d'avoir plusieurs idées que nous avons essayé de réaliser au cours de ces années de recherche.

Le **deuxième chapitre** développe une proposition de deux modèles parallèles PPSO1 et PPSO2 basés sur la méta-heuristique PSO. Ces modèles ont été expérimentés sur une un ensemble de fonctions tests connues dans la littérature pour évaluer leurs performances ; les résultats obtenus ont prouvé leur efficacité en terme de temps de calcul et de qualité de la solution par rapport au PSO classique.

La réalisation de ces deux modèles et leurs expérimentations nous a mené à une réflexion plus poussée, afin de réaliser un modèle parallèle plus performant MPSO, ce dernier a fait l'objet du **troisième chapitre**. L'algorithme MPSO associe le voisinage évolutif au calcul parallèle dans le but d'améliorer la qualité des résultats obtenus des deux modèles PPSO1 et PPSO2. MPSO a été appliqué sur un problème réel de l'optimisation, celui du transport d'électricité ; en optimisant la durée de vie du pylône d'une ligne de transport d'électricité ; l'objectif été de maximiser la résistance à la charge tout en réduisant le coût. Les résultats obtenus du modèle MPSO comparés aux résultats du processus d'optimisation de premier

ordre ANSYS sont satisfaisants en terme de diminution du temps de calcul et de qualité de la convergence.

Une variante de l'algorithme MPSO a été développée dans le **quatrième chapitre,** nommée H-MPSO-SA. Elle a été proposée pour améliorer la qualité de la solution surtout pour les problèmes complexes contenant un grand nombre d'optima locaux (vu la grande capacité de recherche locale de l'algorithme SA). H-MPSO-SA a été testé sur une série de fonctions tests complexes, les résultats obtenus ont amélioré d'avantage la qualité de la solution par rapport à l'algorithme MPSO. H-MPSO-SA a ensuite été testé pour améliorer la mise en œuvre des réseaux de collaboration. Les résultats obtenus ont été satisfaisants.

Perspectives

Les perspectives de ce travail sont nombreuses, tant du point de vue théorique que pratique. L'une des perspectives qui nous semble la plus prometteuse et la plus stimulante est l'étude de l'évolution du modèle parallèle MPSO en l'adossant à une modélisation basée sur le paradigme multiagent.

L'approche consiste à donner aux particules plus d'autonomie, une exécution asynchrone et des capacités d'apprentissage supérieures. L'environnement serait transformé en une ressource plus dynamique et informative.

Dans l'algorithme PSO, les particules dans l'essaim ont une autonomie et une intelligence limitées [64]. La vitesse de la particule dans l'espace du problème est régie par un algorithme central identique pour toutes les particules. Il donne à la particule une flexibilité limitée et une connaissance de l'environnement. Bien que cela rende chaque calcul plus efficace, la performance globale peut être améliorée en ajoutant une sophistication supplémentaire à l'algorithme. Nous estimons qu'élever la particule au statut d'agent améliorera l'efficacité de l'ensemble de l'algorithme, surtout pour le vaste espace de problèmes dotés d'une structure complexe (le modèle MPSO est d'ailleurs dédié à cette catégorie de problèmes).

Nous proposons une nouvelle approche basée sur le modèle MPSO en considérant l'essaim comme un système multiagent.
Un système multiagent (SMA) est composé d'un ensemble d'agents logiciels autonomes capables de réaliser les objectifs souhaités de manière coopérative. Les attributs de base d'un agent qui sont considérés comme typiques sont l'autonomie, l'apprentissage et la coopération [65]. Ces propriétés impliquent que les agents sont capables de s'exécuter indépendamment de tout autre contrôle et éventuellement de manière asynchrone. Ils découvrent les connaissances pertinentes de l'environnement ainsi que d'autres agents susceptibles de les

aider à atteindre les objectifs souhaités par un travail de coopération et en concurrence avec d'autres agents.

Les SMA et l'essaim de particules dans la méthode PSO ont des similitudes:

Dans les deux cas nous avons des approches basées sur des populations qui accomplissent des tâches en coopération. Cependant, les agents se distinguent des particules de manière très spécifique. Nous en explicitons trois raisons pertinentes pour notre approche. Tout d'abord, une particule n'est généralement pas considérée comme autonome car elle ne peut se déplacer que dans l'espace du problème en fonction de l'algorithme principal. Les agents sont quant à eux capables d'explorer l'environnement avec beaucoup plus de flexibilité alors que les particules sont volontairement définies comme moins intelligentes ce qui n'alourdit pas les performances en temps de calcul. La capacité d'apprentissage, qui est une forme d'intelligence, est une composante importante pour les agents. Enfin, PSO applique une exécution synchronisée afin de maintenir la simplicité de la conception. Toutefois, en raison de l'autonomie, des composants d'apprentissage et de coopération, les agents du SMA agissent naturellement exécutés de manière asynchrone. Pour tout cela, il nous semble possible de tirer parti d'une approche combinant le parallélisme et le voisinage évolutif du MPSO avec l'autonomie et l'apprentissage du SMA.

En introduisant l'autonomie et l'apprentissage dans le modèle MPSO, les particules deviennent plus intelligentes et autonomes: elles peuvent atteindre une meilleure performance et une utilisation plus efficaces des ressources. Puisque chaque particule serait maintenant en mesure de tirer parti de son environnement, elle peut demander des informations à ses voisines et mettre à jour son emplacement. Cela économise des ressources de calcul et pourrait accélérer les performances.

Pour ce modèle proposé, basé sur l'algorithme MPOS, chaque particule est considérée comme un agent, le voisinage est un agent. De plus, l'environnement, lui-même modélisé en tant qu'agent, est chargé de fournir aux agents les informations relatives au nombre de voisinages pour décider s'il faut augmenter ou diminuer ce nombre selon les conditions

posées. Nous proposons aussi de contrôler ce nombre de voisinages afin de garder la stabilité de l'algorithme et éviter la convergence prématurée.

Sur la base de cette méthodologie, l'environnement statique est maintenant transformé en un environnement dynamique dans lequel le nombre de voisinage change à chaque itération. Pour éviter que les particules ne se regroupent quelque part (endroit qui peut être un optimum local) la taile du voisinage serait évolutive. L'agent environnement veille à ce que la gestion de la création des voisinages soit effectuée de la manière la plus appropriée.

Les particules simples dans MPSO sont maintenant transformées en agents dans cette nouvelle approche avec autonomie et capacité d'apprentissage. La propriété autonomie permet aux agents de particules d'être plus proactifs en leur permettant d'agir sur leur environnement et changer de voisinages.
Ce mécanisme d'optimisation facilite l'exécution asynchrone ce qui peut améliorer les performances globales de l'optimisation et permettre au modèle MPSO d'être déployé dans des environnements hétérogènes.

En plus de la découverte de son environnement, chaque agent « particule » se voit également attribuer d'autres capacités. Il s'agit notamment de demander aux particules de son voisinage la meilleure solution actuelle pour calculer le meilleur optimum local, renvoyer sa meilleure solution personnelle à ses voisins et demander des informations sur le nombre de voisinages à l'agent d'environnement. Ces méthodes supplémentaires étendent également les fonctionnalités de la particule au-delà d'un simple agent.

En fin de compte, en agissant sur le levier du parallélisme, notre recherche nous a mené vers les systèmes multiagents. Nous prévoyons ainsi explorer la question de l'optimisation par la voie des systèmes distribués. Nous venons d'initier ce travail et en brosser les grandes lignes de modélisation.

Annexes

Sommaire

A.1 Modélisation et implémentation des modèles proposés

La modélisation des modèles parallèles proposés en Java a été effectuée comme suit : l'application a été découpée en trois couches, chaque couche est représenté par un paquetage Java :

- La couche présentation : cette couche représente l'interaction avec l'utilisateur final : elle permet d'intercepter les interactions de l'utilisateur avec l'application et les acheminer vers la couche au-dessous (traitement).
- La couche traitement : cette couche représente le cœur (noyau) du programme. Pour PPSO1, le cœur du programme : permet d'exécuter le traitement sur un espace global qui a été subdivisé en sous-espaces, chaque sous-espace contient une topologie formée par un groupe de particules. Le traitement consiste en la création d'un ensemble de traitements parallèles « threads Java », chacun de ces traitements est un algorithme PSO exécuté sur un groupe de particules se positionnant dans des voisinages différents.

  Pour PPSO2, le cœur du programme : permet de lancer des PSO parallèles, un thread par voisinage, ce dernier est représenté par un groupe de particules dans une zone de l'espace de recherche sans qu'il ai de communication entre les différents groupes.
- La couche accès aux données : cette couche permet la sauvegarde de la configuration (la taille de l'essaim, la taille de l'espace de recherche, la disposition des particules, la valeur des paramètres PSO), afin de les envoyer à la couche traitement en cas de besoin.

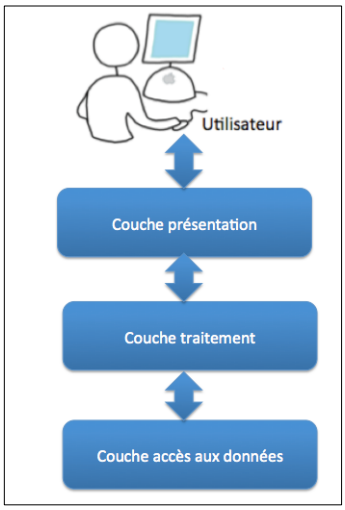


Figure A- 1 Représentation des composantes d'une application.

A.1.1 Diagramme de classe

Le coeur faisant tourné le projet est situé dans le package de la couche traitement. Il contient les classes de base comme représenté sur le diagramme ci-dessous (voir la Figure A-2).

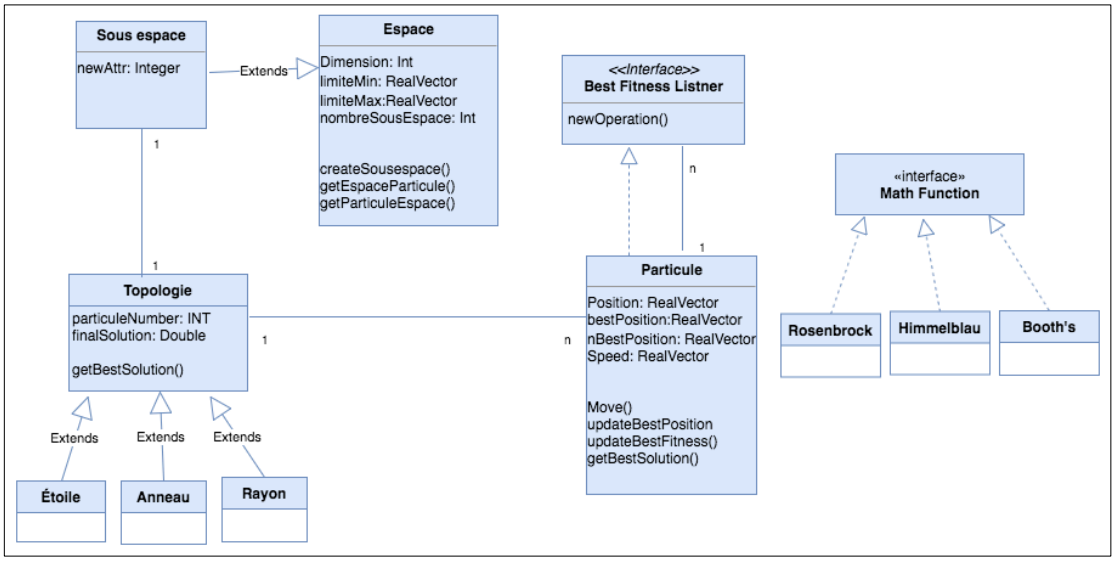


Figure A- 2 Digramme de classe du modèle PPSO1.

A.1.2 Diagramme des cas d'utilisation

Le diagramme est composé d'un seul acteur qui est l'utilisateur qui accède à l'interface PSO afin d'effectuer les différentes tâches susmentionnées ci-dessous :

- **Initialiser PSO :** l'utilisateur accède à l'interface PSO et ajuste les différents paramètres PSO selon ses besoins, à savoir : le nombre des particules, le nombre des itérations, la fonction à optimiser, le nombre de voisinages, la dimension et la taille de l'espace de recherche.
- **Sauvegarder Espace :** l'utilisateur peut sauvegarder à chaque fois la configuration initiale avec tous les paramètres utilisés afin de pouvoir les réutiliser en cas de besoin.
- **Lancer la recherche :** l'utilisateur peut lancer la recherche de l'optimum de la fonction qu'il a utilisé.
- **Arrêter la recherche :** l'utilisateur à la possibilité d'arrêter la recherche si la solution trouvée jusqu'à cet instant lui paraît optimale et convient ses besoins.
- **Enregistrer les résultats :** l'utilisateur peut enregistrer les résultats obtenus lors de ses tests.
- **Consulter les résultats :** l'utilisateur peut accéder au fichier contenant les résultats afin de les comparer ou de les utiliser selon ses besoins.

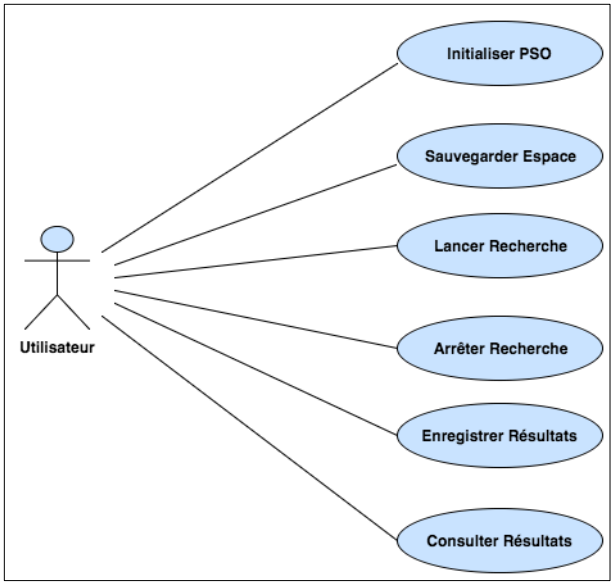


Figure A- 3 Digramme des cas d'utilisation du programme PPSO1.

A.1.3 Interface Homme Machine

Afin de faciliter l'utilisation de nos modèles, nous avons crée une interface graphique qui permet à l'utilisateur de tester le programme avec aisance.

La Figure A-4 représente l'interface graphique avant toute modification. Les valeurs indiquées sont celles codées en brut dans le programme. Elles peuvent être modifiées en choisissant les paramètres selon le besoin de l'utilisateur à savoir : le nombre de particules, le nombre d'itérations, la dimension de l'espace de recherche, la taille des voisinages, la fonction à optimiser, etc.

- Le bouton "Regenerate Particles" permet d'initialiser les positions des particules dans l'espace de recherche.
- Le bouton "Next iteration" permet de faire le suivi du déplacement des particules à chaque itération.
- Le bouton "Results" permet d'afficher les résultats finaux de chaque voisinage et la meilleure solution obtenue.

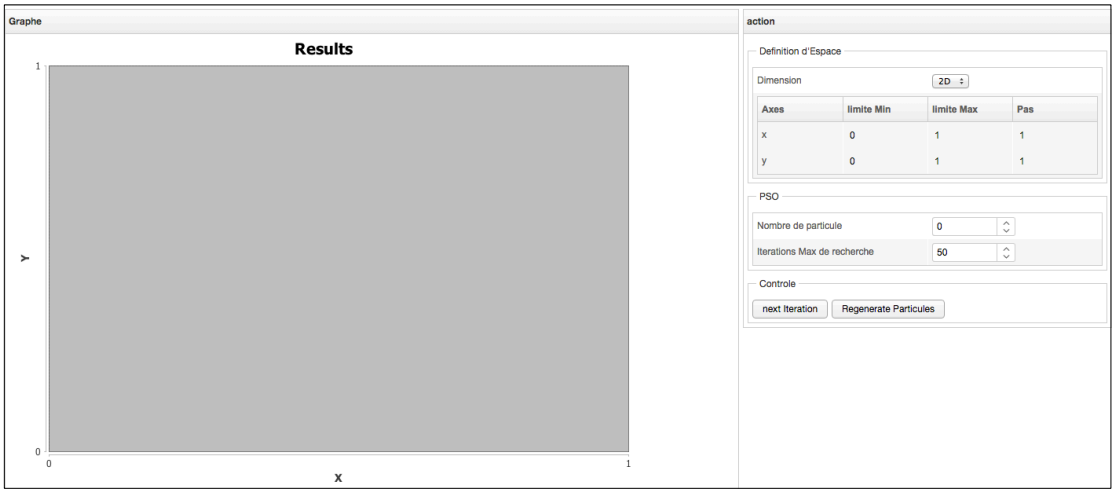


Figure A- 4 Capture de l'interface graphique avant la configuration.

La Figure A-5 représente l'interface graphique après la configuration des paramètres et création des différents voisinages.

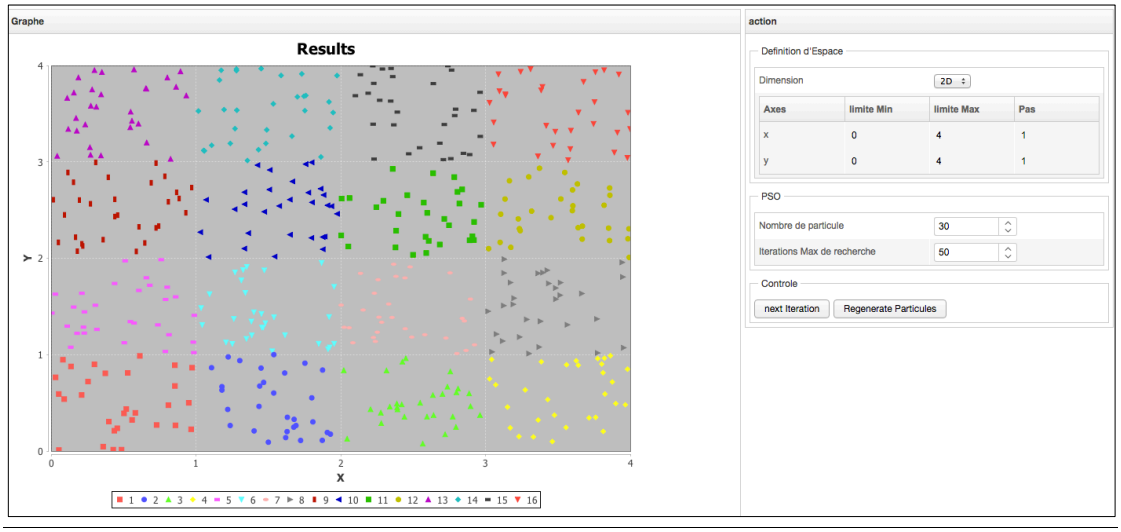


Figure A- 5 Capture de l'interface graphique après configuration.

La Figure A-6 représente l'interface graphique en cours d'exécution, les particules changent de voisinage et se dirigent vers l'optimum au fil des itérations.

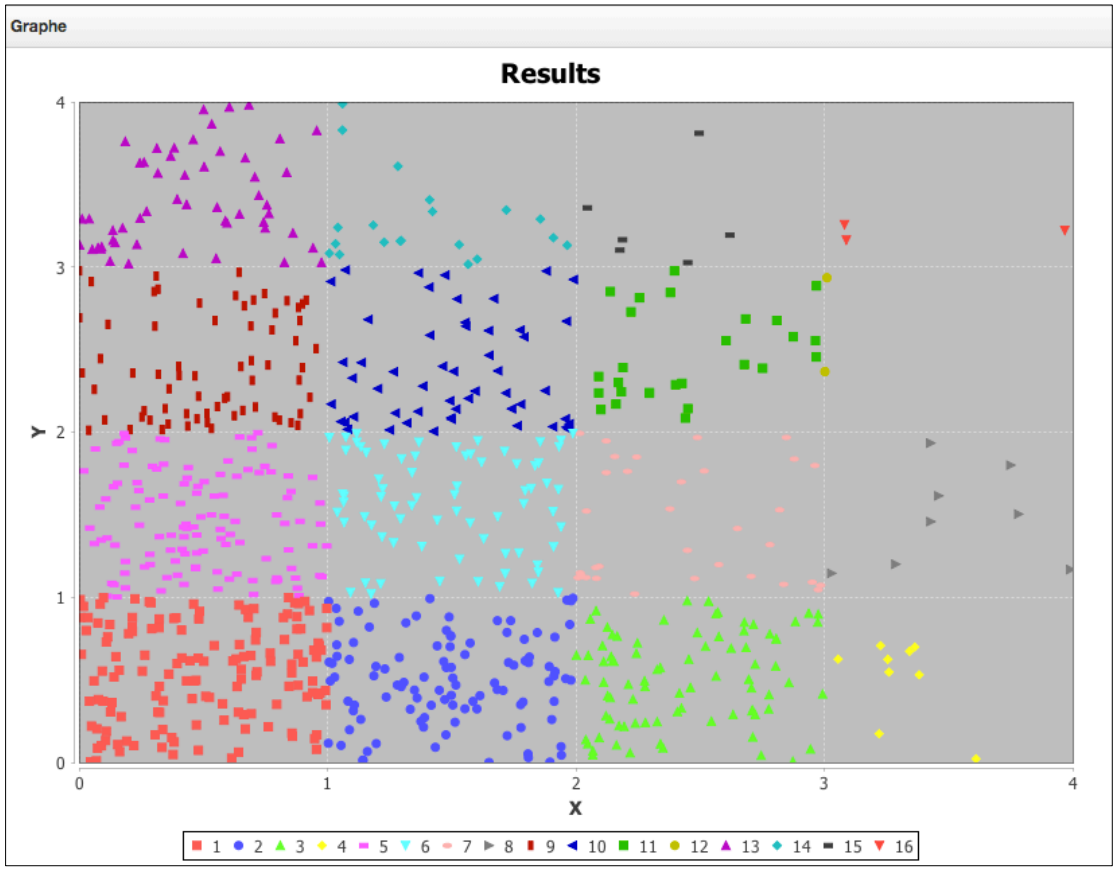


Figure A- 6 Capture de l'interface graphique en cours d'exécution.

La Figure A-7 représente l'interface graphique pour la partie des résultats, il y à l'ensemble des informations relatives à l'exécution: les paramètres PSO configurés par l'utilisateur ainsi que les résultats de l'exécution (le temps d'exécution, la meilleure solution obtenue, la meilleure position, et l'emplacement de la solution obtenue).

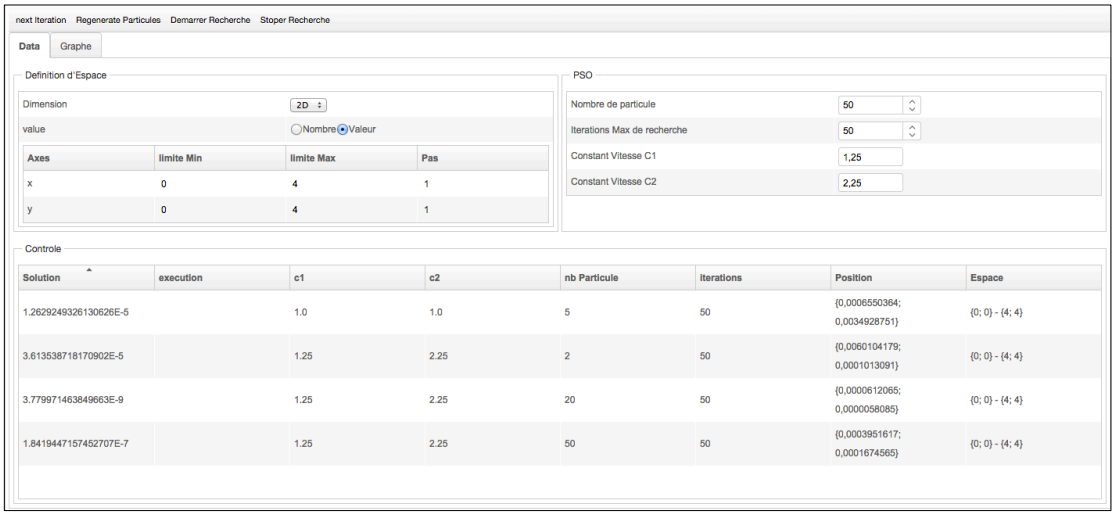


Figure A- 7 Capture de l'interface graphique contenant le résultat de l'exécution.

A.1.4 Technologies utilisées

L'application est réalisée en utilisant un ensemble de technologies :

- Le langage Java/JEE qui est une spécification pour la plate-forme Java d'Oracle, destinée aux applications d'entreprise.
- Le framework open source « ZK » pour la couche présentation proposant une interaction utilisateur (UI) riche, en se basant sur le pattern MVVM « Model View ViewModel ». ZK permet tout autant de définir rapidement des interfaces graphiques via une syntaxe XML ou un éditeur Wysiwyg que de manipuler directement les objets en Java.
- L'outil « Apach Maven Project » : est un outil destiné à construire les projets informatique pour obtenir un livrable (ou un « artefact ») tel qu'un exécutable, un JAR ou encore un WAR (destiné à être déployé sur un serveur d'applications). Nous l'avons utilisé pour la construction du projet et la récupération des librairies externes.
- Le conteneur web « Apach Tomcat » est une implémentation open source d'un conteneur web qui permet donc d'exécuter des applications web reposant sur les technologies servlets et JSP. Nous l'avons utilisé pour l'exécution web de l'application.

- Le Pattern « Listner des Bean Java » pour la communication entre les particules. Les Listners Java est une technologie qui permet l'écoute de certains événements dans une application. Lorsqu'un objet Listener est entrain d'écouter un événement et que ce dernier se déclenche, alors le listener s'active : en général, une de ses méthodes particulière est invoquée, avec en paramètre un objet portant les informations sur cet événement. Le principe de fonctionnement est donc celui du callback, on enregistre des listeners auprès du serveur d'application. Ces objets doivent implémenter des interfaces standard, fournies par l'API Servlet, et être déclarés dans le descripteur d'une application web : le fichierweb.xml. Pour la bonne communication de l'information entre les différents membres d'un groupe et qui forment une topologie (Ring, Star, ..), nous avons implémenté un mécanisme basé sur le Pattern « Listner des Bean Java », qui permet de partager le changement du Best de chaque particule avec les autres membres du groupe selon la topologie utilisée.
- L'API «ThreadPoolExecutor» pour le traitement parallèle. Comme la création des Threads est couteuse, nous avons pensé à utiliser l'API «ThreadPoolExecutor» afin de réaliser les traitements parallèles. Cette approche a en particulier la caractéristique de la réutilisation des Threads crées.

A.2 Fonctions tests mathématiques

Dans cette section, une présentation des fonctions tests utilisées dans nos expérimentations pour évaluer les performances de nos algorithmes proposés [66].

A.2.1 Fonction de Rosenbrock

La fonction de Rosenbrock, aussi appelée la fonction « vallée » ou « banane », est un problème de test populaire pour les algorithmes d'optimisation par gradient. Il est présenté dans le graphique ci-dessous sous sa forme en 2 dimensions (Voir Figure A-8).

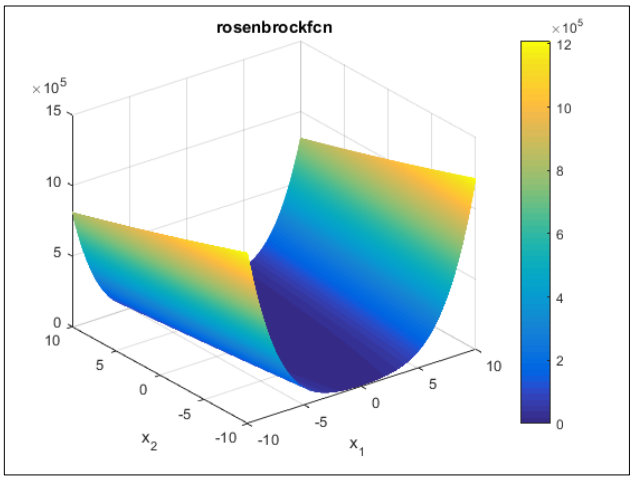


Figure A- 8 Fonction de Rosenbrock en 2 dimensions.

La fonction est unimodale et le minimum global se situe dans une vallée étroite et parabolique. Sa formule mathématique est définie comme suit:

$$f(x) = \sum_{i-1}^{n-1} \left[100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2 \right]$$

Avec: $$-30 \le x_i \le 30$$

Le minimum global est obtenu au point (x, y)= (1,1) pour lequel la fonction vaut 0.

A.2.2 Fonction d'Himmelblau

La fonction d'Himmelblau est une fonction de test multimodale conçue pour l'évaluation de la qualité des méthodes d'optimisation. Sa formule mathématique est définie comme suit:

$$f(x,y) = (x^2 + y - 11)^2 + (x + y^2 - 7)^2$$

Avec: $$-5 \le x_i, y_i \le 5$$

La fonction comporte 4 minima globaux obtenus aux points $(x,y) = (3,2)$, $(x,y) = (-2.805118, 3.283186)$, $(x,y) = (-3.779310, -3.283186)$, $(x,y) = (3.584458, -1.848126)$ pour lesquels la fonction vaut zéro. La Figure A-9 est une représentation de la fonction d'Himmelblau en 2 dimensions.

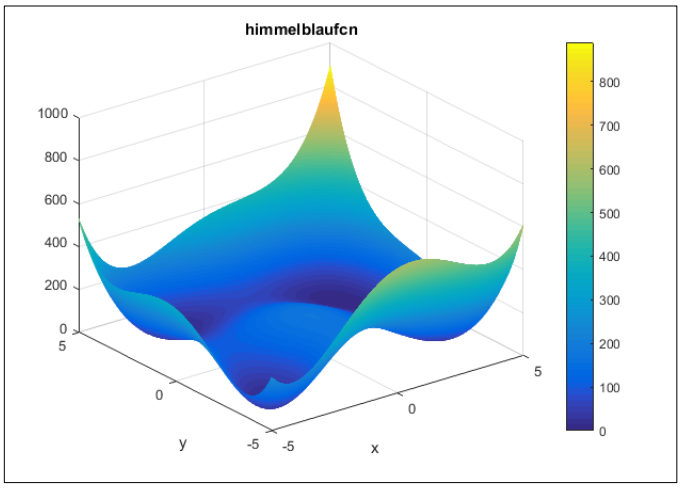


Figure A- 9 Fonction d'Himmelblau en 2 dimensions.

A.2.3 Fonction de Beale

La fonction de Beale est une fonction mathématique multimodale, avec des pics aigus aux angles du domaine de recherche. Sa formule mathématique est définie comme suit:

$$f(x,y) = (1.5 - x + xy)^2 + (2.25 - x + xy^2)^2 + (2.625 - x + xy^3)^2$$

Avec: $$-5 \le x_i, y_i \le 5$$

Le minimum global est obtenu au point (x, y)= (3,0.5) pour lequel la fonction vaut 0. La Figure A-10 est une représentation de la fonction de Beale en 2 dimensions.

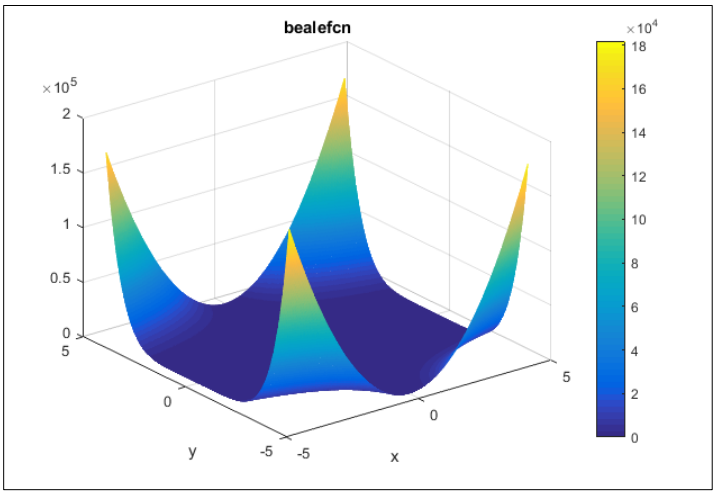


Figure A- 10 Fonction de Beale en 2 dimensions.

A.2.4 Fonction d'Easom

La fonction d'Easom est une fonction de test mathématique continue et multimodale conçue pour l'évaluation de la qualité des méthodes d'optimisation. Sa formule mathématique est définie comme suit:

$$f(x,y) = -\cos(x)\cos(y)\exp(-((x-\pi)^2 + (y-\pi)^2))$$

Avec: $$-100 \leq x_i, y_i \leq 100$$

Le minimum global est obtenu au point (x, y)= (π,π) pour lequel la fonction vaut -1. La Figure A-11 est une représentation de la fonction d'Easom en 2 dimensions.

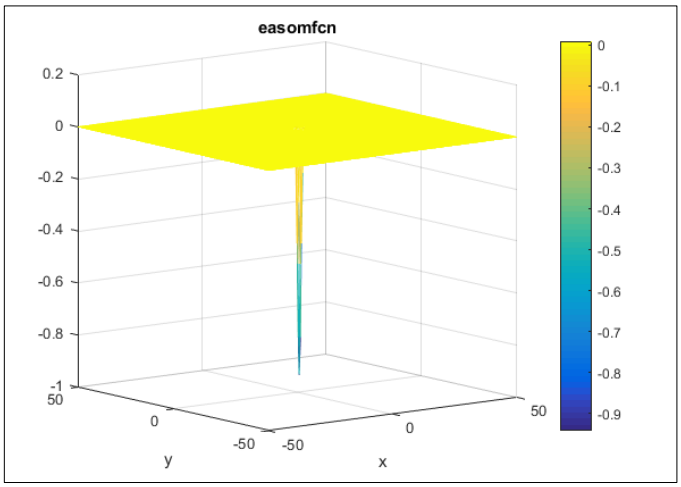


Figure A- 11 Fonction d'Easom en 2 dimensions.

A.2.5 Fonction de Sphère

La fonction de Sphère appelée aussi « paraboloïde» est une fonction de test mathématique convexe, continue et unimodale conçue pour l'évaluation de la qualité des méthodes d'optimisation. Sa formule mathématique est définie comme suit:

$$f(x) = \sum_{i:1}^{n} x_i^2$$

Avec: $$-5.12 \leq x_i, y_i \leq 5.12$$

La valeur du minimum global de la fonction est égale zéro. La Figure A-12 est une représentation de la fonction de Sphère en 2 dimensions.

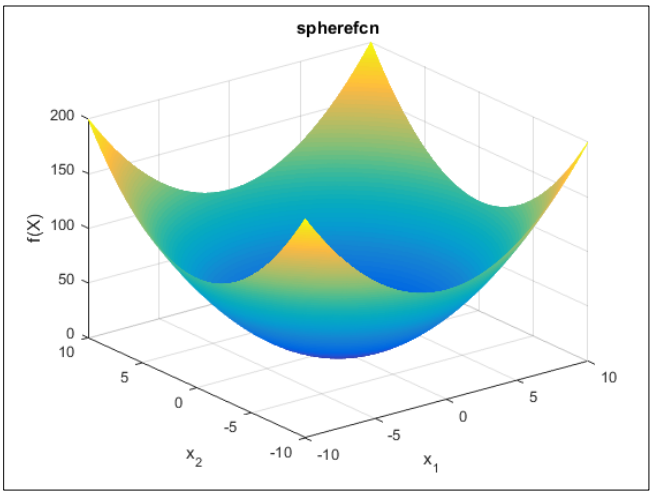


Figure A- 12 Fonction de Sphère en 2 dimensions.

A.2.6 Fonction de Rastring

La fonction de Rastring est une fonction convexe, continue, multimodale et à dimension élevée conçue pour l'évaluation de la qualité des méthodes d'optimisation. Sa formule mathématique est définie comme suit:

$$f(x,y) = 0.5 + \frac{\sin^2(x^2+y^2)^2 - 0.5}{(1+0.001(x^2+y^2))^2}$$

Avec: $$-100 \le x_i \le 100$$

La fonction de Rastring contient un ensemble de minima locaux, mais un seul minimum global, sa valeur est égale zéro. La Figure A-13 est une représentation de la fonction de Rastring en 2 dimensions.

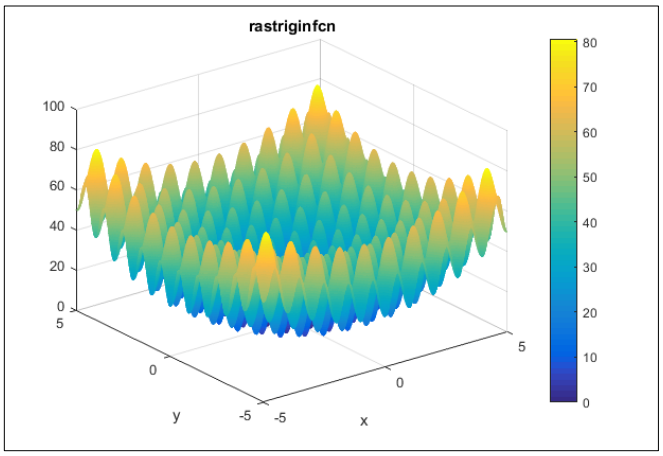


Figure A- 13 Fonction de Rastring en 2 dimensions.

A.2.7 Fonction de Griewank

La fonction de Griewank est une fonction de test mathématique convexe, continue et unimodale, comporte plusieurs minima locaux étalés sur l'ensemble de l'espace der recherche et sont uniformément distribués. Sa formule mathématique est définie comme suit:

$$f(x) = 1 + \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right)$$

Avec: $$-600 \le x_i, y_i \le 600$$

La valeur du minimum global de la fonction est égale zéro. La Figure A-14 est une représentation de la fonction Griewank de en 2 dimensions.

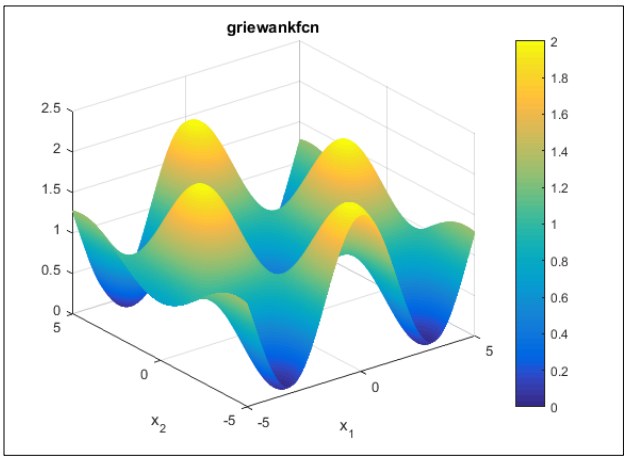


Figure A- 14 Fonction de Griewank en 2 dimensions.

A.2.8 Fonction d'Ackley

La fonction d'Ackley est une fonction de test mathématique convexe, continue et multimodale. Sa formule mathématique est définie comme suit:

$$f(x) = \sum_{i=0}^{n} \alpha_i \chi_{Ai}(x)$$

Avec: $-5 \leq x_i \leq 5$

Le minimum global est obtenu au point (x, y)= (0,0) pour lequel la fonction vaut 0. La Figure A-15 est une représentation de la fonction d'Ackley de en 2 dimensions.

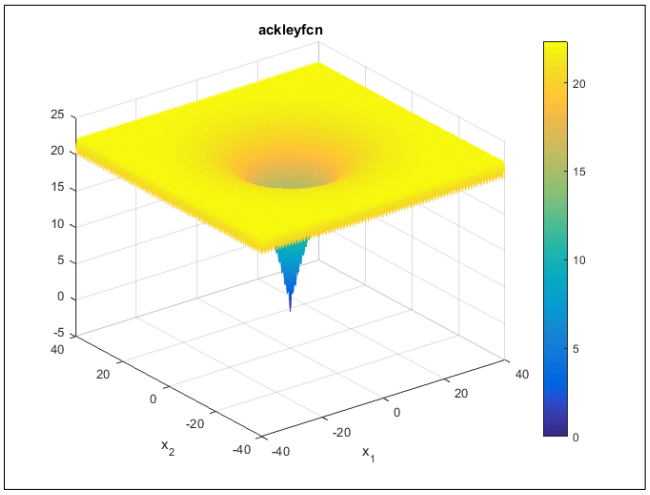


Figure A- 15 Fonction d'Ackley en 2 dimensions.

A.2.9 Fonction de Matyas

La fonction de Matyas est une fonction de test mathématique convexe, continue et unimodale. Sa formule mathématique est définie comme suit:

$$f(x,y) = 0.26(x^2 + y^2) - 0.48xy$$

Avec: $-10 \leq x_i, y_i \leq 10$

Le minimum global est obtenu au point (x, y)= (0,0) pour lequel la fonction vaut 0. La Figure A-16 est une représentation de la fonction de Matyas de en 2 dimensions.

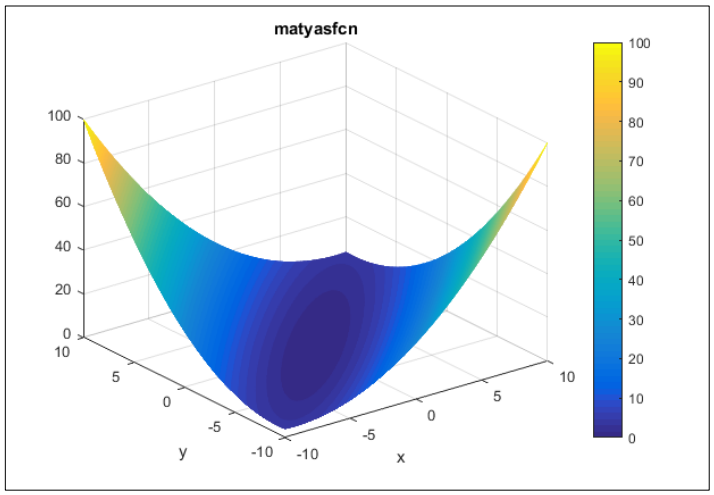


Figure A- 16 Fonction de Matyas en 2 dimensions.

A.2.10 Fonction de Booth

La fonction de Booth est une fonction de test mathématique convexe, continue et unimodale. Sa formule mathématique est définie comme suit:

$$f(x,y) = (x+2y-7)^2 + (2x+y-5)^2$$

Avec: $$-10 \le x_i, y_i \le 10$$

Le minimum global est obtenu au point (x, y)= (1,3) pour lequel la fonction vaut 0. La Figure A-17 est une représentation de la fonction de Booth de en 2 dimensions.

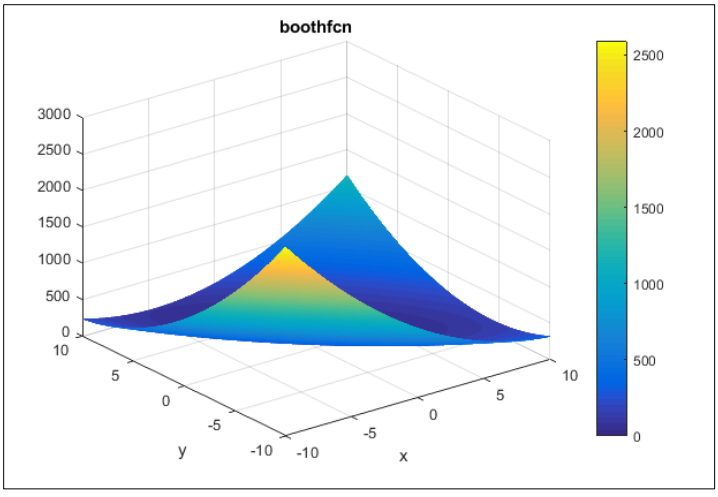


Figure A- 17 Fonction de Booth en 2 dimensions.

A.2.11 Fonction de McCormick

La fonction de McCormick est une fonction de test mathématique convexe, continue et multimodale. Sa formule mathématique est définie comme suit:

$$f(x,y) = \sin(x+y) + (x-y)^2 - 1.5x + 2.5y + 1$$

Avec:

$$-1.5 \le x_i \le 4$$
$$-3 \le y_i \le 4$$

Le minimum global est obtenu au point (x, y)= (-0.54719,-1.54719) pour lequel la fonction vaut -1.9133. La Figure A-18 est une représentation de la fonction de McCormick de en 2 dimensions.

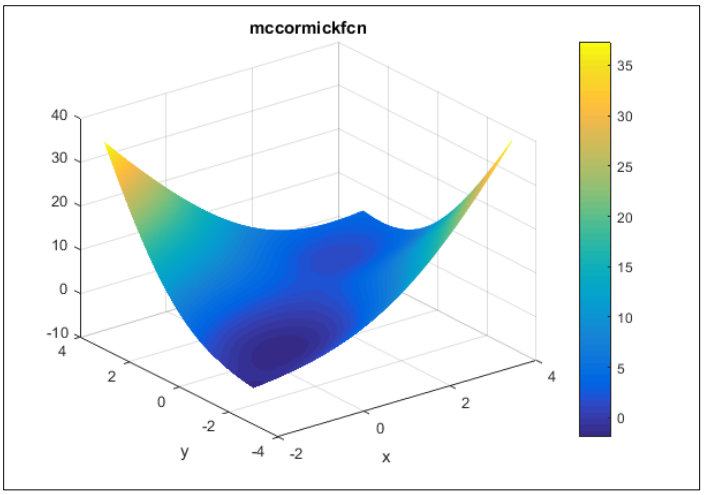


Figure A- 18 Fonction de McCormick en 2 dimensions.

A.2.12 Fonction de Three-Hump Camel

La fonction de Three-Hump Camel est une fonction de test mathématique convexe, continue et multimodale. Sa formule mathématique est définie comme suit:

$$f(x,y) = 2x^2 - 1.05x^4 + \frac{x^6}{6} + xy + y^2$$

Avec:

$$-5 \le x_i, y_i \le 5$$

Le minimum global est obtenu au point (x, y)= (0,0) pour lequel la fonction vaut 0. La Figure A-19 est une représentation de la fonction de Three-Hump Camel de en 2 dimensions.

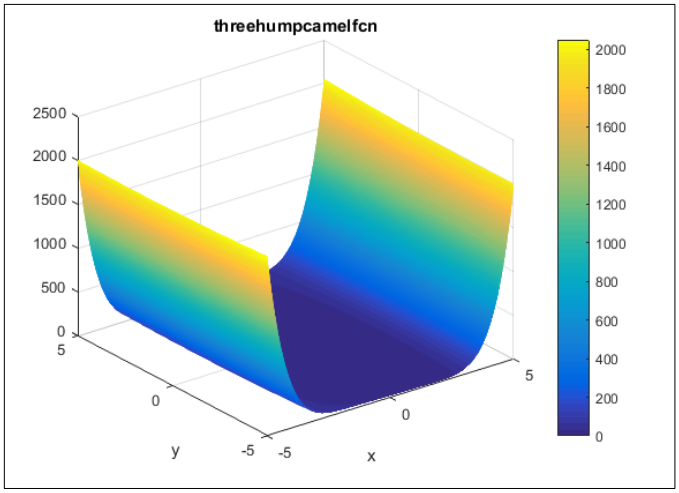


Figure A- 19 Fonction de Three-Hump Camel en 2 dimensions.

A.2.13 Fonction de Goldstein-Price

La fonction de Goldstein-Price est une fonction de test mathématique convexe, continue et multimodale. Sa formule mathématique est définie comme suit:

$$f(x,y)=\left[1+(x+y+1)^2(19-14x+3x^2-14y+6xy+3y^2)\right]$$
$$\left[30+(2x-3y)^2(18-32x+12x^2+48y-36xy+27y^2)\right]$$

Avec: $-2 \le x_i, y_i \le 2$

Le minimum global est obtenu au point (x, y)= (0,-1) pour lequel la fonction vaut 3. La figure A.20 est une représentation de la fonction de Goldstein-Price de en 2 dimensions.

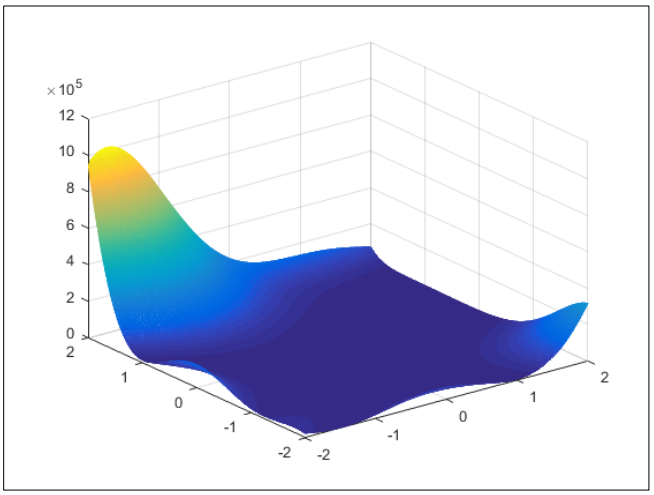


Figure A- 20 Fonction de Goldstein-Price en 2 dimensions.

A.2.14 Fonction de Cross-in-tray

La fonction de Cross-in-tray est une fonction de test mathématique convexe, continue et multimodale. Sa formule mathématique est définie comme suit:

$$f(x,y) = -0.0001\left[\left|\sin x \sin y \exp\left(\left|100 - \frac{\sqrt{x^2+y^2}}{\pi}\right|\right)\right|+1\right]^{0.1}$$

Avec: $-10 \le x_i, y_i \le 10$

La fonction comporte 4 minima globaux obtenus aux points $(x,y) = (\pm 1.349406685353340, \pm 1.349406608602084)$ pour lesquels la fonction vaut -2.06261218. La Figure A-21 est une représentation de la fonction de Cross-in-tray de en 2 dimensions.

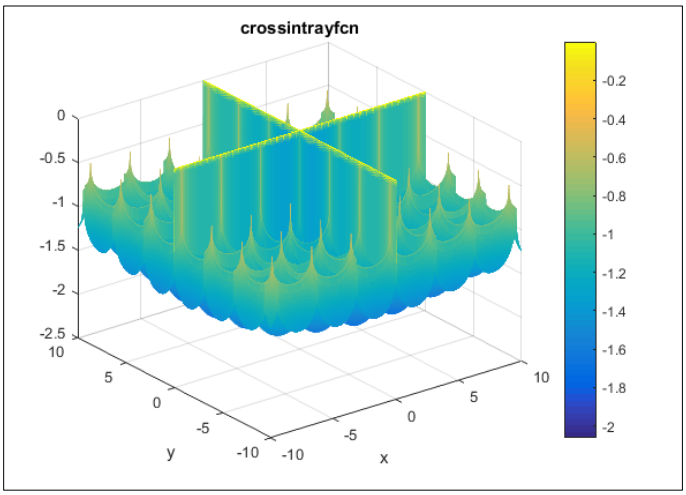


Figure A- 21 Fonction de Cross-in-tray en 2 dimensions.

A.2.15 Fonction de Holder-Table

La fonction de Holder-Table est une fonction de test mathématique convexe, continue et multimodale. Sa formule mathématique est définie comme suit:

$$f(x,y) = -\left|\sin x \cos y \exp\left(\left|1 - \frac{\sqrt{x^2+y^2}}{\pi}\right|\right)\right|$$

Avec: $-10 \le x_i, y_i \le 10$

La fonction comporte 4 minima globaux obtenus aux points $(x,y) = (\pm 8.05502, \pm 9.66459)$ pour lesquels la fonction vaut -19.2085. La Figure A-22 est une représentation de la fonction de Holder-Table de en 2 dimensions.

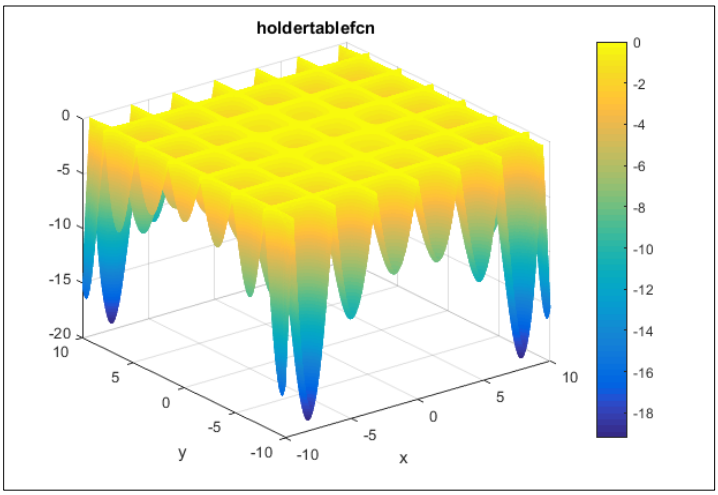


Figure A- 22 Fonction de Holder-Table en 2 dimensions.

A.2.16 Fonction de Schaffer N.1

La fonction de Schaffer N.1 est une fonction de test mathématique convexe, continue et unimodale. Sa formule mathématique est définie comme suit:

$$f(x,y) = 0.5 + \frac{\sin^2(x^2+y^2)^2 - 0.5}{(1+0.001(x^2+y^2))^2}$$

Avec: $$-100 \le x_i \le 100$$

Le minimum global est obtenu au point (x, y)= (0,0) pour lequel la fonction vaut 0. La Figure A-23 est une représentation de la fonction de Schaffer N.1 de en 2 dimensions.

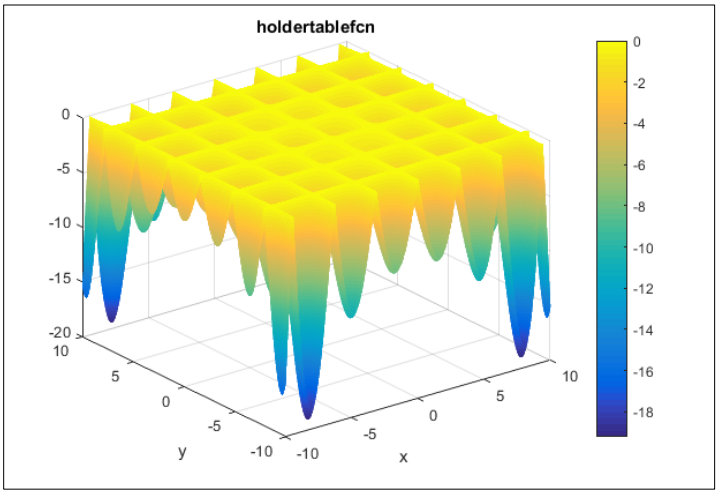


Figure A- 23 Fonction de Schaffer N.1 en 2 dimensions.

Bibliographie

[1] Y. Cooren,: “Perfectionnement d'un algorithme adaptatif d'Optimisation par Essaim Particulaire. Applications en génie médical et en électronique”. Thèse de Doctorat, Université de Paris 12 Val de Marne, France, 2008.

[2] N. Elhami,: “Contribution aux méthodes hybrides d’optimisation heuristiques: Distribution et application à l’interopérabilité des systèmes d’information”. Thèse de Doctorat, Université Mohammed V Rabat, Maroc & Université de Rouen, France, 2013.

[3] K. Byung-I et G. Alan,: "Parallel asynchronous particle swarm optimization" International Journal For Numerical Methods In Engineering, vol. 67, pp. 578-595, 2006.

[4] Shiyuan Jin, Damian Dechev, Zhihua Qu,: “Parallel Particle Swarm Optimization (PPSO) on the Coverage Problem in Pursuit-Evasion Games”. Conference: Proposed for presentation at the 20th High Performance Computing Symposium (HPC 2012), Orlando, FL. 2012.

[5] P. Rabanal, I. Rodríguez et F. Rubio: “Parallelizing Particle Swarm Optimization in a Functional Programming Environment”. Algorithms2014 : vol. 7, pp. 554–581, 2014.

[6] Arionde Campo Aurora T.R.Pozo Elias P.Duarte,: “Parallel multi-swarm PSO strategies for solving many objective optimization problems”. Journal of Parallel and Distributed Computing. Vol 126, pp.13-33, 2019.

[7] Mickeal J.Flynn “Some Computer Organizations and Their Effectiveness”. IEEE Trans. Comput., Vol. C-21, p. 948, 1972.

[8] Laurence Viry et Violaine Louvet Ecole d’Automne “Genéralités sur le parallélisme” . Ecole d’Automne ‘Informatique Scientifique pour le Calcul’, 2008.

[9] GENE Amdahl, “Validity of the Single Processor Approach to Achieving Large-Scale Computing Capabilities”. AFIPS Conference Proceedings, (30), p. 483-485,1967.

[10] John L.Gustafson et Edwin H. Barsis, “Reevaluating Amdahl's Law” . Communications of the ACM. pp.532-533, 1988.

[11] Alan H. Karp et Horace P. Flatt,. "Measuring Parallel Processor Performance". Communications of the ACM. 33 (5): pp.539–543, 1990.

[12] Soniya Lalwani, Harish Sharma, Suresh Chandra Satapathy, Kusum Deep et Jagdish Chand Bansal,: “A Survey on Parallel Particle Swarm Optimization Algorithms”. Arabian Journal for Science and Engineering, 2019.

[13] https://neos-guide.org/content/optimization-tree-alphabetical. (Dernière visite 17/04/2019).

[14] Abbas El Dor. "Perfectionnement des algorithmes d’optimisation par essaim particulaire : applications en segmentation d’images et en électronique". Thèse de doctorat.. Université Paris-Est, 2012.

[15] J. Kennedy et R. Eberhart,: “Particle Swarm Optimization,”. Proceedings of the IEEE International Joint Conference on Neural Networks, IEEE Press, vol. 8, no. 3, pp. 1943–1948, 1995.

[16] A. Ratnaweera, SK. Halgamuge, HC. Watson,: “Self-organising hierarchical particle swarm optimizer with time-varying acceleration coefficients”. IEEE Trans Evol Comput. Vol. 8, N° 3, pp. 240-255, 2004.

[17] I-H. Kuo, S-J. Horng, T-W. Kao, T-L. Lin, P. Fan,: “An Efficient Flow-Shop Scheduling Algorithm Based on a Hybrid Particle Swarm Optimization Model”. New Trends in Applied Artificial Intelligence. pp. 303-312, Springer. 2007.

[18] K. E. Parsopoulos et M. N. Vrahatis,: “Recent approaches to global optimization problems through particle swarm optimization”. Natural Computing: an international journal, 1(2-3), pp. 235-306, 2002.

[19] M. E. Hyass et P. Hyass,: “Good Parameters for Particle Swarm Optimization”. Laboratories Technical Report no. HL1001. 2010.

[20] Guillaume CALAS,: "Optimisation par essaim de particules". Spécialisation Sciences Cognitives et Informatique Avancée (SCIA) École d'Ingénieurs en Informatique EPITA. France. 2009.

[21] Pant, M., Thangaraj, R., & Abraham, A,: "Particle swarm optimization using adaptive mutation". 19th International Conference on Database and Expert Systems , Washington, DC, USA, pp. 519-523, 2008.

[22] Nguyen, U. Q., Hoai, N. X., McKay, R., & Tuan, P. M,: "Initializing PSO with randomized low-discrepancy sequences: the comparative results". Proceedings of the IEEE Congress on Evolutionary Computation pp. 1985-1992, 2007.

[23] K. Zielinski et R. Laur,: "Stopping Criteria for Differential Evolution in Constrained Single-Objective Optimization". Advanced in Differential Evolution, the series Studies in Computational Intelligence Vol. 143, pp. 111-138 Springer, Berlin Heidelberg. 2008.

[24] Long, H.X.; Li, M.Z.; Fu, H.Y.: "Parallel quantum-behaved particle swarm optimization algorithm with neighborhood search". International Conference on Oriental Thinking and Fuzzy Logic, pp. 479–489, 2016.

[25] Pant, M., Thangaraj, R., Grosan, C., & Abraham, A,: "Improved particle swarm optimization with low-discrepancy sequences". IEEE Cong. on Evolutionary, Hong Kong, pp. 3011-3018, 2008.

[26] Peng, Y.; Peng, A.; Zhang, X.; Zhou, H.; Zhang, L.; Wang, W.; Zhang, Z.: "Multi-core parallel particle swarm optimization for the operation of inter-basin water transfer-supply systems". Water Resour. Manag. 31(1), 27–41, 2017.

[27] Chang Zhang , Zhiwei Ni , Zhangjun Wu et Lichuan Gu,: "A novel swarm model with quasi-oppositional particle". International Forum on Information Technology and Applications, pp. 325 –330, 2009.

[28] Yuan, S.; Zhao, L.; Mu, B.: "Parallel cooperative co-evolution based particle swarm optimization algorithm for solving conditional nonlinear optimal perturbation". International Conference on Neural Information Processing, pp. 87–95, 2015.

[29] Y. Shi et R.C. Eberhart,: “Parameter selection in particle swarm optimization”. Proceedings of the Seventh Annual Conference on Evolutionary Programming, pp. 591-600, 1998.

[30] M. Clerc et J. Kennedy,: “The particle swarm - explosion, stability, and convergence in a multidimensional complex space”. IEEE Trans. Evolutionary Computation, 6(1):pp. 58-73, 2002.

[31] Russ C. Eberhart et Y. Shi,: “Comparing inertia weights and constriction factors in particle swarm optimization”. Proceedings of the Congress on Evolutionary Computing, pp. 84-89, 2000.

[32] R. Eberhart, P. Simpson, and R. Dobbins. “Computational Intelligence PC Tools”. AP Professional, 1996.

[33] K. Deep, J. C. Bansal,: “Hybridization of particle swarm optimization with quadratic approximation. OPSEARCH”. Vol. 46, N° 1, pp. 3-24, 2009.

[34] J. Barrera, C. A.C. Coello,: “Limiting the velocity in particle swarm optimization using a geometric series”. Genetic And Evolutionary Computation Conference, Proceedings of the 11th Annual conference on Genetic and evolutionary computation, pp. 1739-1740, 2009.

[35] X. Cai, Y.Tan,: “A study on the effect of vmax in particle swarm optimization with high dimension”. International Journal of Bio-Inspired Computation (IJBIC). Vol. 1, N°. 3, pp. 210 - 216, 2009.

[36] B. Bochnek et P. Fory’s, : “Structural optimization for post buckling behavior using particle swarms". Struct Multidisc Optim. pp. 521-531, 2006.

[37] R. Mendes,: “Population Topologies and Their Influence in Particle Swarm Performance”. Thèse de doctorat. Université de Minho, Portugal, 2004.

[38] S. Kirkpatrick, C. D. Gelatt Jr, M. P. Vecchi,: “Optimization by Simulated Annealing”. Science, New Series, Vol. 220, Issue 4598, pp. 671-680, 1983.

[39] Siarry P., Drryfus G. , “Application of Physical Methods to the Computer-Aided Design of Electronic Circuits”, J. Phys. Lett. 45, L 39, 1984.

[40] Cerny V,: “A thermodynamical approach to the travelling salesman problem: an efficient simulated algorithm”. Journal of Optimization Theory and Applications, no 45, pp. 41-51, 1985.

[41] Amira Gherboudj, “Méthodes de résolution de problèmes difficiles académiques”. Thèse de doctorat. Université de Constantine2, Algérie. 2013.

[42] Metropolis N., Rosenbluth A., Rosenbluth M., Teller A., Teller E., “Equation of state calculations by fast computing machines”. Journal of Chemical Physics, vol. 21, pp. 1087-1092, 1953.

[43] Autin Baptiste: “Les métaheuristiques en optimisation combinatoire”. Mémoire de fin d’études, Conservatoire National Des Arts et Métiers, Paris. 2006.

[44] J. Chang, S. Chu, J. Roddick et J. Pan,: “A Parallel Particle Swarm Optimization Algorithm With Communication Strategies”. Journal of Information Science and Engineering. 2005.

[45] Gerhard Venter et Jaroslaw Sobieszcanski,: “Parallel particle swarm optimization algorithm accelerated by asynchronous evaluations”. J. Aerosp. Comput. Inf. Commun. 3(3), pp. 123–137, 2006.

[46] Byung-II Koh, Alan D. George, Raphael Haftka et Benjamin J Fregly,: “Parallel asynchronous particle swarm optimization”. Communications in Numerical Methods in Engineering 67(4) pp. 578-595, 2006.

[47] K. Byung-I et G. Alan,: "Parallel asynchronous particle swarm optimization". International Journal For Numerical Methods In Engineering, vol. 67, pp. 578-595, 2006.

[48] Andrew W. McNabb ; Christopher K. Monson ; Kevin D. Seppi,: “Parallel PSO using MapReduce”. IEEE Congress on Evolutionary Computation. 2007.

[49] Hari Prasain ; Girish Kumar Jha ; Parimala Thulasiraman ; Ruppa Thulasiram,: “A parallel Particle swarm optimization algorithm for option pricing”. IEEE International Symposium on Parallel & Distributed Processing, Workshops and Phd Forum (IPDPSW). 2010.

[50] Narjiss Dali et Sadok Bouamama,: “GPU-PSO : Parallel Particle Swarm Optimization approaches on Graphical Processing Unit for Constraint Reasoning: Case of Max-CSPs”. 19th International Conference on Knowledge Based and Intelligent Information and Engineering Systems. Procedia Computer Science 60 (2015) pp.1070 – 1080, 2015.

[51] M.Zemzami, N.Elhami, M.Itmi et N.Hmina,: “Parallèlisation de la Méthode PSO: Découpage de l’espace et traitement par lot des particules”. International Workshop on New Services and Networks (WNSN’16). Khouribga. Maroc. 2016.

[52] M.Zemzami, N.Elhami, M.Itmi et N.Hmina,: “A New Parallel Approach For The Exploitation Of The Search Space Based On PSO Algorithm”. 4th International Colloquium in Information Science and Technology (CIST’16). Tanger. Maroc. 2016.

[53] M. I. Aouad,; “Conception d’algorithmes hybrides pour l’optimisation de l’énergie mémoire dans les systèmes embarqués et de fonctions multimodales”. Thèse de Doctorat, Université Henri Poincaré- Nancy 1, France. 2011.

[54] M.Zemzami, A.Makhloufi, N.Elhami, M.Itmi et N.Hmina,: "Electrical Power Transmission Optimization based on a New Version of PSO Algorithm". (Publié le 22/02/17 DOI: 10.21494/ISTE.OP.2017.0127).

[55] M.Zemzami, N.Elhami, M.Itmi et N.Hmina,: "A modified Particle Swarm Optimization algorithm linking dynamic neighborhood topology to parallel computation". International Journal of Engineering and Technology (IJATCSE) Volume 8, N2 pp 112-118. 2019.

[56] ANSYS Guide 2015, ANSYS Structural Analysis Guide, 2015.

[57] M.Zemzami, A.Elhami, A.Makhloufi, N.Elhami, M.Itmi et N.Hmina,: "Applying a new parallelized version of PSO algorithm for electrical power transmission". International Conference on Materials Engineering and Nanotechnology (ICMEN’17). 12-14 Mai 2017 Kuala Lumpur, Malysia. IOP Conference Series: Materials Science and Engineering (Online ISSN: 1757-899X; Print ISSN: 1757-8981) 2017.

[58] Standard Computer Dictionary - A Compilation of IEEE Standard Computer Glossaries, 1990.

[59] Elmir B. et Bounabat B.: “E-Service Interoperability Measurement within Business Collaboration Networks,” Proceedings of MICS, 2010.

[60] Daclin N., Chen D., and Vallespir B. : “Enterprise interoperability measurement-Basic concepts.,”. In EMOI-INTEROP, 2006.

[61] Kasunic M., Anderson W.: “Measuring Systems Interoperability: Challenges and Opportunities.” Technical Note CMU/SEI-2004-TN-003.Carnegie Mellon University, Pittsburgh, 2004.

[62] M.Zemzami, N.Elhami, M.Itmi et N.Hmina,: " Interoperability optimization using a modified PSO algorithm". International Journal of Engineering and Technology (IJATCSE) Volume 8, N2 pp 101-107, 2019.

[63] M.Zemzami, N.Elhami, M.Itmi et N.Hmina,: "An evolutionary hybrid algorithm for complex optimization problems". International Journal of Engineering and Technology (IJATCSE) Volume 8, N 2, pp 126-133, 2019.

[64] Ahmad, Raheel, Lee, Yung-Chuan, Rahimi, Shahram et Gupta, Bidyut,. "A Multi-Agent Based Approach for Particle Swarm Optimization." IEEE International Conference on Integration of Knowledge Intensive Multi-Agent Systems, 2007. KIMAS 2007, 267-271. doi: 10.1109/KIMAS. 2007.

[65] Miguel KPAKPO “Une approche de gestion de la maintenance de parcs eoliens centrée sur les systèmes multiagents”. Thèse de Doctorat, Université de Normandie, INSA Rouen, France. 2018.

[66] [Benchmarkfcns] Mazhar.ansari.ardeh “ BenchmarkFcns”. http://benchmarkfcns.xyz/fcns. (Dernière visite 19/04/2019).