République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Batna 2

Faculté des mathématiques et d'informatique

Département d'informatique

THESE DE DOCTORAT

En vue de l'obtention du diplôme de Doctorat en Informatique

Option : Systèmes et réseaux informatiques

Intitulée

OPTIMISATION MULTI-OBJECTIF : ÉTUDES DE CAS

Présentée et soutenue par :
Aouadj Wafa

Le : 23/09/2021

Devant le jury composé de :

Pr. Benharzallah Saber	**Professeur**	**Université Batna 2**	**Président**
Pr. Seghir Rachid	**Professeur**	**Université Batna 2**	**Rapporteur**
Pr. Bennoui Hammadi	**Professeur**	**Université de Biskra**	**Examinateur**
Dr. Laboudi Zakaria	**MCA**	**Université de OEB**	**Examinateur**
Pr. Hedjazi Djalal	**Professeur**	**Université Batna 2**	**Examinateur**
Dr. Abdessemed Mohamed Rida	**MCA**	**Université Batna 2**	**Invité**

À toute ma famille formidable

À la mémoire de mon père

Remerciements

الحمد لله

Je remercie mon directeur de thèse le professeur Seghir Rachid de m'avoir fait confiance durant ces années de travail. Je le remercie pour ses conseils avisés, sa gentillesse, sa disponibilité permanente et pour les nombreux encouragements qu'il m'a prodigués.

Je tiens à remercier le professeur Benharzallah Saber d'avoir accepté d'être président du jury. Je remercie également les membres du jury : le professeur Bennoui Hammadi, le docteur Laboudi Zakaria et le professeur Hedjazi Djalal d'avoir accepté d'examiner et assister à la présentation de ce travail.

Je remercie notre invité le docteur Abdessemed Mohamed Rida d'avoir accepté notre invitation et d'avoir contribué au travail présenté dans cette thèse. Je le remercie aussi pour ses remarques précieuses et ses conseils avisés.

J'exprime ma gratitude au staff du Centre des technologies de l'information Université Batna 2 pour leur soutien matériel généreux et continu.

Merci aussi au staff de la bibliothèque centrale de l'université Batna 1 qui nous a aidé pour travailler durant les vacances d'été.

Je tiens à remercier toutes mes amies et collègues qui ont été là pour m'écouter durant les durs moments, plus particulièrement Wassila, Nassima, Hadjer et Ikram.

Enfin, tous les mots ne suffisent pas pour remercier ma **mère**, mes frères : Imed, Okba, Amer et Youcef et ma sœur Khaoula qui ont su me soutenir, me supporter, m'encourager. . . pendant toute la durée de ma thèse.

Résumé

Cette thèse porte sur l'étude d'un problème d'optimisation multi-objectif dans le domaine de la robotique en essaim. La problématique est de concevoir un ou plusieurs modèles comportementaux d'un ensemble d'agents-robots ayant comme tâche le regroupement d'objets, tout en satisfaisants deux objectifs : la maximisation de la qualité du tas formé et la minimisation de l'énergie consommée pour accomplir la tâche. La motivation pour étudier ce cas, en particulier, est de contribuer à ouvrir la voie à la conception d'un système de robots en essaim conscient de l'énergie qu'il consomme tout en fournissant une bonne qualité en termes de la tâche de formation de tas accordée. La contribution comporte trois volets principaux. Le premier concerne la re-modélisation du problème habituel de regroupement d'objets à un seul objectif pour traiter les deux objectifs contradictoires. Un algorithme d'optimisation multi-objectif a été utilisé afin de concevoir l'ensemble des comportements individuels d'agents-robots et analyser le comportement global de l'essaim. Contrairement à d'autres solutions proposées pour résoudre le problème de regroupement d'objets, l'énergie consommée est estimée pendant la phase de conception. En plus, une distinction en termes de la consommation énergétique est faite entre les modèles des comportements qui fournissent la même qualité de regroupement. Enfin, l'énergie consommée pour accomplir la tâche est moins variée d'une expérience à une autre. Dans le deuxième, on s'intéresse à l'analyse du modèle comportemental choisi dans différentes circonstances ; en utilisant ce modèle, la robustesse, la scalabilité et la flexibilité du système sont aussi examinées. La supériorité de l'approche proposée est prouvée en comparant deux modèles comportementaux bi-objectif à un autre mono-objectif. Tandis que, le troisième volet se focalise sur la proposition d'une variante de l'algorithme d'optimisation basé sur l'enseignement-apprentissage pour résoudre ce problème. L'algorithme a montré des performances prometteuses face à un problème multi-objectif discret à grande échelle.

Mots clefs

Optimisation multi-objectif, optimisation à grande échelle, problème discret, robotique en essaim, modèle comportemental, robot autonome mobile, optimisation basée sur l'enseignement-apprentissage.

Abstract

This thesis deals with the study of a multi objective optimization problem in the field of swarm robotics. The problem is to design one or more behavioral models of a set of agent-robots having as a task the grouping of objects, while satisfying two objectives: maximizing the quality of the formed heap and minimizing the energy consumed to accomplish the task. The motivation to study this case, in particular, is to help pave the way for the design of a swarm robotic system aware of the consumed energy while providing a good quality in terms of the given task. Three contributions have been proposed in this thesis. In the first part, the usual problem of object clustering with a single objective is remodeled to deal with the two contradictory objectives. A multi-objective optimization algorithm has been used to design the set of individual behaviors of agent-robots and to analyze the overall behavior of the swarm. Unlike other solutions proposed to solve the object clustering task, the energy consumed is estimated during the design phase. Also, a distinction is made between models of behaviors that provide the same quality of clustering. Finally, the energy consumed to accomplish the task is less varied from one experiment to another. In the second part, a behavioral model is analyzed under different circumstances; using this model, the robustness, scalability and flexibility of the system are also examined. The superiority of the proposed approach is proved by comparing two bi-objective behavioral models against another mono-objective model. In a third part, a variant of teaching-learning based optimization algorithm is proposed to solve the object clustering problem. The algorithm has shown promising performance against the presented discrete large-scale multi-objective problem.

Keywords

Multi-objective optimization, large-scale optimization, discrete problem, swarm robotics, behavioral model, autonomous mobile robot, stochastic approach, teaching-learning based optimization algorithm.

الملخص

تركز هذه الرسالة على دراسة حالة من مشاكل التحسين متعددة الأهداف في مجال روبوتات السرب. تكمن المشكلة في تصميم نموذج واحد أو أكثر من النماذج السلوكية لمجموعة من الروبوتات التي تتمثل مهمتها في تجميع الكائنات، مع تحقيق هدفين: زيادة جودة الكومة المتكونة وتقليل الطاقة المستهلكة لإنجاز المهمة. الدافع لدراسة هذه الحالة، على وجه الخصوص، هو المساعدة في تمهيد الطريق لتصميم نظام روبوتات سرب مدرك للطاقة التي يستهلكها مع توفير جودة جيدة في إنجاز المهمة المعينة. تتكون المساهمة من ثلاثة مكونات رئيسية. من ناحية أولى، تم إعادة تصميم المشكلة المعتادة المتمثلة في تجميع الكائنات بهدف واحد للتعامل مع الهدفين المتعارضين. تم استخدام خوارزمية تحسين متعددة الأهداف لتصميم مجموعة من سلوكيات الروبوت الفردية وتحليل السلوك العام للسرب. على عكس الحلول الأخرى المقترحة لحل مشكلة الطاقة داخل سرب، يتم تقدير الطاقة المستهلكة خلال مرحلة التصميم، ويتم التمييز بين نماذج السلوك التي توفر نفس جودة التجميع، والطاقة المستهلكة لإنجاز المهمة أقل تباينا من تجربة إلى أخرى. من ناحية ثانية، تم تحليل النموذج السلوكي في ظروف مختلفة؛ باستخدام هذا النموذج، تم أيضًا فحص متانة النظام وقابلية توسيعه ومرونته. ولإثبات تفوق النهج المقترح، تم مقارنة نموذجين سلوكيين ثنائي الهدف مع آخر بهدف واحد. في الجزء الثالث، تم اقتراح متغير من خوارزمية التحسين القائمة على التعليم والتعلم لحل هذه المشكلة. أظهرت الخوارزمية أداءً واعدًا في مواجهة هذه المشكلة متعددة الأهداف، منفصلة وواسعة النطاق.

الكلمات المفتاحية

التحسين متعدد الأهداف، التحسين على نطاق واسع، المشكلة المنفصلة، سرب الروبوتات، النموذج السلوكي، روبوت-وكيل محمول و مستقل، التحسين القائم على التعليم والتعلم.

Sommaire

Liste des figures

Liste des tableaux

Liste des algorithmes

Liste des abréviations

A	ACO	Ant Colony Optimization
	A-R	Agent-Robot
C	CB	Capacité de Batterie
	CD	Crowding Distance
	CMA-ES	Covariance Matrix Adaptation Evolution Strategy
D	DE	Distance Euclidienne
	DLM-TLBO	Discrete Large-scale Multi-objective Teaching-Learning-Based Optimization
E	EAPSO	Energy Aware Particle Swarm Optimization
	EPSO	Energy-based Particle Swarm Optimization
F	F_q	Fonction objectif de la Qualité
	F_e	Fonction objectif de l'Energie
G	GA	Genetic Algorithms
	GRASP	Greedy Randomized Adaptative Search Procedure
H	HTL-MOPSO	Hybrid Teaching Learning-based Multi-Objective Particle Swarm Optimization
	HTLBO-HS	Hybrid Teaching-Learning Based Optimization with Harmony Search
I	INM-TLBO	Multi-Objective Individualized-Instruction TLBO
	LETLBO	Teaching–Learning-Based Optimization with Learning Experience
L	LR	Learning Rate
	LSO	Large-Scale Optimization
M	MBFO	Multi-objective Bacterial foraging Optimization
	MC-BO	Modèle Comportemental Bi-Objectif
	MC-MO	Modèle Comportemental Mono-Objectif
	MSSA	Multi-objective Salp Swarm Algorithm
	MOABC	Multi-Objective Artificial Bee Colony
	MOACO	Multi-Objective Ant Colony Optimization
	MOAIS	Multi-Objective Artificial Immune System
	MO-BBO	Multi-Objective Biogeography Based Optimization
	MOCS	Multi-Objective cuckoo search
	MODA	Multi-Objective Dragonfly Algorithm
	MODE	Multi-Objective Differential Evolution
	MOEP	Multi-Objective Evolutionary Programming
	MO-GLS	Multi-Objective Guided Local Search
	MO-GWO	Multi-Objective Gray Wolf Optimization
	MOGP	Multi-Objective Genetic Programming
	MOLSO	Multi-Objective Large-scale optimization

	MOSA	Multi-Objective Simulated annealing
	MOTLBO	Multi-Objective Teaching-Learning-Based Optimization
	MOTS	Multi-Objective Tabu Search algorithm
	MO-VNS	Multi-Objective Variable Neighborhood Search
N	NBI	Normal Boundary Intersection
	NC	Normal Constraint
	NDS	Non-dominated Sorting
	NSGA-II	Non-dominated Sorting Genetic Algorithm -II
O	OCT	Object Clustering Task
	OMO	Optimisation Multi-Objectif
P	POMO	Problème d'Optimisation Multi-Objectif
	PSO	Particle Swarm Optimization
R	RE	Robotique Evolutionnaire
S	SA	Simulated annealing
	SA-MTLBO	Self-Adaptive Multi-objective Teaching-Learning-Based Optimization
	SMES	Surrogate-assisted Multi-objective Evolution Strategy
	SMR	Système Multi-Robot
	SRE	Système de Robots en essaim
T	TLBO	Teaching-Learning Based Optimization
V	VEGA	Vector Evaluated Genetic Algorithm
W	WS	Weighted Sum
2	2S-TLBO	2 Stages Teaching-Learning Based Optimization

Introduction générale

Récemment, il y a eu un intérêt accru pour l'optimisation qui date déjà de plusieurs décennies. Le besoin de concevoir et d'utiliser des solutions ou des modèles optimaux est l'un des sujets clés. De ce fait, la résolution de problèmes d'optimisation est devenue indispensable que ce soit dans la conception en ingénierie ou d'autres domaines. L'importance de l'optimisation est traduite par la tendance et la croissance rapide des travaux utilisant ou bien développant de nouvelles méthodes. Souvent, dans de nombreux problèmes du monde réel, plusieurs objectifs doivent être gérés simultanément, appelés Problèmes d'Optimisation Multi-Objectif (POMO). Contrairement au problème mono-objectif qui a une seule solution optimale, un problème d'optimisation multi-objectif possède un ensemble de solutions optimales appelé Front de Pareto. Cette discipline joue un rôle de première importance dans la conception en ingénierie, et d'une manière générale dans la gestion et la prise de décision. Elle s'agit de trouver les meilleurs compromis entre les objectifs en conflit. En résumé, le domaine de l'OMO décrit l'art et la manière scientifique de prendre de telles décisions [1].

La plupart de ces problèmes d'OMO appartiennent à la classe des problèmes NP-difficiles où il n'existe pas d'algorithme générique fournissant des solutions optimales en un temps polynomial. La complexité de ces problèmes est estimée en fonction de la taille du problème et du nombre des objectifs à optimiser. Différentes méthodes mathématiques et algorithmes stochastiques sont proposés afin de résoudre ce type de problèmes. Dans la catégorie d'algorithmes stochastiques à base de population, on trouve ceux qui sont inspirés de la biologie, de la physique, de la société humaine, etc.

Parmi les algorithmes inspirés de la société humaine, il y a l'algorithme Teaching-Learning Based Optimization (TLBO) proposé dans sa forme mono-objectif [2, 3] en 2011. Il imite le processus d'enseignement-apprentissage classique. Différemment aux autres algorithmes à base de population, TLBO se compose de deux phases : la phase d'enseignement (Teaching phase) et la phase d'apprentissage (Learning phase). Chaque individu de la population est considéré comme un apprenant. Il apprend des connaissances dispensées par un enseignant, et d'autres en interagissant avec d'autres apprenants par le biais de discussions de groupe, de présentations et de communications formelles.

Le développement de méthodes de résolution ainsi que leurs applications dans divers domaines de la vie réelle sont en continuelle croissance. Comme domaine d'application de l'optimisation en générale et de l'optimisation multi-objectif en particulier, la conception de système robotique est l'un des domaines les plus prometteurs. La conception de robots a débuté très tôt vers le 20 nième siècle, mais avec l'accroissement des besoins humains dans le temps, elle est devenue de plus en plus complexe et chère. L'apparition de la robotique en essaim a permet d'envisager l'utilisation de robots peu sophistiqués, moins couteux et plus facile à concevoir. Un ensemble restreint de règles simples, appelé modèle comportemental, peut guider les agents-robots de l'essaim vers un comportement global généralement plus sophistiqué.

Plusieurs scénarios ont été traités dans la robotique en essaim, parmi eux, le problème de regroupement d'objets, appelé aussi formation en tas [4], est choisi comme cas d'étude. Un phénomène de formation d'un cimetière est observé chez plusieurs espèces de fourmis [5, 6, 7]. Sans avoir transporté les fourmis mortes et les déchets loin du nid, la moisissure va se développer dans le nid [7]. Dans quelques heures, les ouvriers peuvent rassembler des cadavres distribués aléatoirement. La tâche de regroupement d'objets est inspirée de ce phénomène. Un cluster final émerge de l'assemblage et du désassemblage continus des sous-clusters. L'emplacement de la collection d'objets n'est pas prédéfini, les robots doivent décider où le tas sera formé.

Problématique

D'un point de vue : « problème traité »

Quoique les travaux déjà réalisés sur le regroupement d'objets soient variés en termes de scénarios, métriques, méthodes de conception et analyse, le point commun entre ces travaux c'est que la tâche de regroupement d'objets était traitée comme un problème à objectif unique ; il s'agissait de trouver un modèle comportemental réussissant la formation d'un tas avec une qualité maximale. Cependant, si le système doit opérer d'une manière ininterrompue dans un endroit sans possibilité de rechargement de batteries (zone désertique ou une planète comme Mars), la consommation d'énergie devient un véritable problème et le choix d'actions consommant moins d'énergie devient une nécessité pour aller jusqu'au bout de la tâche à accomplir. Sinon, la mission risque d'être interrompue en raison du nombre croissant des robots qui se trouveront hors service. Parmi les lacunes de conception d'un système de robots en essaim qui ne prend pas en considération l'énergie consommée lors de l'accomplissement de la tâche, nous retrouvons ce qui suit :

(1) L'énergie consommée par les robots n'est pas prise en considération pendant la phase de conception. Plus tard, dans la phase de mise-en-œuvre, les batteries intégrées doivent répondre aux exigences du modèle comportemental adopté. Ce qui limite le choix de la technologie ;
(2) Aucune distinction n'est faite entre les modèles comportementaux qui fournissent la même qualité de regroupement, car tous les comportements sont considérés comme similaires en termes de consommation énergétique, ce qui n'est pas le cas ;
(3) L'énergie consommée pour accomplir la tâche ciblée est très variée d'une expérience à une autre. Parce que plusieurs actions, qui consomment différentes quantités d'énergie, sont considérées comme équivalentes et le choix entre ces actions se fait de manière aléatoire.

D'un autre côté, les travaux proposés pour résoudre le problème de consommation d'énergie au sein d'un essaim de robots sont dirigés vers le rechargement d'énergie, l'homéostasie énergétique collective ou bien l'optimisation de l'énergie consommée. Dans cette dernière classe, les travaux sont orientés vers la décomposition temporelle de la tâche ciblée, l'affectation de sous-tâches à des sous-groupes de robots ou bien la limitation d'activités si le robot souffre d'un manque énergie. Le problème d'énergie n'a pas été considéré comme un problème de choix de modèle comportemental.

D'un point de vue : « méthode de résolution »

Le point commun entre la majorité des algorithmes à base de population est les paramètres spécifiques à chaque algorithme. Ces paramètres jouent un rôle important dans l'algorithme. L'ajustement de ces derniers est un processus qui peut parfois être difficile et compliqué, ce qui affecte directement l'efficacité de l'algorithme.

L'optimisation basée sur le processus enseignement-apprentissage est une technique qui ne nécessite aucun réglage de paramètre spécifique de l'algorithme. Elle a montré son efficacité appréciable pour

résoudre différents types de problèmes malgré sa simplicité et sa facilité à décrire et à implémenter [8]. Depuis sa proposition en 2011, les variantes de cet algorithme sont généralement dédiées aux problèmes discrets ou bien multi-objectif. Une seule variante a été proposée pour résoudre les problèmes à grande échelle. Quoique les résultats prouvent la capacité de TLBO à résoudre les problèmes mono-objectif à grande échelle [3]. A nos connaissances, aucune variante n'a été proposée pour résoudre les problèmes multi-objectif à grande échelle.

Objectifs de la thèse

(1) Remodeler le problème de regroupement d'objets et trouver les compromis (modèles de comportements) permettant au système robotique de minimiser la consommation d'énergie et de maximiser la qualité de formation du tas.
(2) Proposer une nouvelle méthode d'optimisation d'énergie consommée basée sur le choix de comportement adéquat et considérer le problème d'énergie comme un problème de choix du modèle comportemental ;
(3) Répondre à la question comment concevoir les comportements des agents-robots pour que le comportement global émerge tout en satisfaisants deux objectifs en conflit ;
(4) Savoir répondre à la question : c'est quoi la relation entre le choix de comportement face à chaque situation, l'énergie totale consommée par le système d'un côté et la qualité de formation du tas d'un autre côté ?
(5) Vérifier la supériorité de la modélisation en un problème bi-objectif sur l'ancienne approche de résolution du problème dans sa forme mono-objectif ;
(6) Évaluer les compromis trouvés en examinant les trois propriétés principales d'un essaim : la scalabilité, la flexibilité et la robustesse ;
(7) Proposer une variante de l'algorithme d'optimisation basée sur l'enseignement-apprentissage pour les problèmes multi-objectif discrets à grande échelle. Et l'implémenter pour améliorer la qualité des solutions obtenues par l'algorithme évolutionnaire déjà utilisé.

Plan de la thèse

Le travail rapporté dans cette thèse est organisé en six chapitres résumés ci-dessous.

Chapitre 1 couvre les concepts de base de l'optimisation multi-objectif. Les types de problèmes d'optimisation et les différentes méthodes de résolution sont passés en revue. Aussi les domaines d'application de l'optimisation multi-objectif les plus importants sont présentés dans ce chapitre.

Chapitre 2 focalise sur l'optimisation basée sur le processus enseignement-apprentissage. Le principe de base de l'algorithme a été présenté, aussi les améliorations, les hybridations les plus importantes et les différentes variantes proposées pour les problèmes multi-objectif sont passées en revue.

Chapitre 3 passe en revue le domaine de la robotique en essaim, la source d'inspiration, les notions de base pour comprendre ce domaine, les propriétés du système en essaim. Ensuite, on se focalise sur le problème traité dans ce travail en présentant les travaux proposés pour résoudre le problème d'énergie et le problème de regroupement d'objets.

Chapitre 4 présente la première partie de notre travail proposé dans ce contexte. Le problème de regroupement d'objets bi-objectif a été décrit en présentant les deux fitness de qualité et de consommation d'énergie. La structure des agents-robots est aussi présentée en décrivant en détail les modules de perception, de comportements de base et de contrôle. Le chapitre contient aussi les résultats

obtenus par l'algorithme évolutionnaire multi-objectif NSGA-II (Non-dominating Sorting Genetic Algoirthm II).

Chapitre 5 s'intéresse à l'évaluation d'un modèle comportemental bi-objectif parmi les solutions trouvées. La scalabilité, la flexibilité et la robustesse du système sont sujettes d'investigation dans un premier lieu. Ensuite, une comparaison entre deux modèles comportementaux bi-objectif et un autre mono-objectif est présentée afin de prouver la supériorité de l'approche proposée.

Chapitre 6 présente l'algorithme proposé, basé sur l'enseignement-apprentissage. Les résultats obtenus en utilisant cet algorithme, pour résoudre le problème remodelé, sont présentés et comparés à ceux obtenus par NSGA-II et la supériorité de l'algorithme proposé a été prouvée.

Cette thèse a donné lieu à une publication et une communication :

[9] W. Aouadj, M.-R. Abdessemed et R. Seghir, «a Reliable Behavioral Model, Optimizing Energy Consumption and Object Clustering Quality by Naïve Robots,» *International journal of swarm intelligence research, scheduled in* vol. 12, n°4, 2021.

[10] W. Aouadj, M.-R. Abdessemed et R. Seghir, «Discrete Large-scale Multi-Objective Teaching-Learning-Based Optimization Algorithm,» *The 4th International Conference on Networking, Information Systems & Security, KENITRA, Morocco,* 1-2 April 2021.

Chapitre 1: L'optimisation multi-objectif

1.1. Introduction

Dans de nombreux problèmes du monde réel, plusieurs objectifs doivent être gérés simultanément, appelés Problèmes d'Optimisation Multi-Objectifs (POMO). Contrairement au problème mono-objectif qui a une seule solution optimale, un POMO possède un ensemble de solutions optimales appelé Front de Pareto [11, 12, 13]. L'optimisation simultanée de plusieurs fonctions « objectifs », généralement en conflit, est dite optimisation multi-objectif (OMO) ou optimisation vectorielle [14, 11]. Cette discipline joue un rôle de première importance dans la conception en ingénierie, et d'une manière générale, dans la gestion et la prise de décision où des spécialistes du domaine d'application ciblé définissent des compromis entre des objectifs de conception en conflit. En somme, le domaine de l'OMO décrit l'art et la manière scientifique de prendre de telles décisions [1]. Souvent, les applications du monde réel ont plusieurs objectifs à optimiser, qui s'opposent. La plupart de ces problèmes d'OMO appartiennent à la classe des problèmes NP-difficiles où il n'existe pas d'algorithme générique fournissant des solutions optimales en un temps polynomial. La complexité de ces problèmes est estimée en fonction de la taille du problème et du nombre d'objectifs à optimiser. Récemment, il y a eu un intérêt remarquable pour l'OMO qui date déjà de plusieurs décennies. Le développement de méthodes de résolution de ce type de problèmes ainsi que leurs applications dans divers domaines de la vie réelle sont en continuelle croissance.

Le reste de ce chapitre est organisé comme suit. La section 1.2 présente une brève définition de l'optimisation. La section 1.3 présente les notions clés de l'optimisation multi-objectif. La section 1.4 discute le processus de résolution d'un POMO. Les approches de résolution sont classées selon le point de vue décideur dans la section 1.5. Tandis que les stratégies de résolution sont classées selon le point de vue du concepteur dans la section 1.6. La section 1.7 discute les différentes méthodes et algorithmes proposés pour résoudre un POMO. Parmi ces algorithmes, les algorithmes basés sur la population sont présentés dans la section 1.8. La section 1.9 résume quelques domaines d'application de l'OMO.

1.2. L'optimisation

Le processus de trouver un ensemble de solutions optimales, parmi toutes les solutions possibles pour un problème donné, est appelé optimisation. Les problèmes d'optimisation (PO) sont classés selon différents critères. Un problème d'optimisation est dit linéaire si toutes les fonctions objectif et contraintes sont linéaires sinon il est dit non-linéaire. Alors qu'un PO est classé selon le type de variables en : continu si ses variables sont continuées ou bien discret si les variables sont du type entier. Les problèmes d'optimisation ayant un seul objectif sont dits mono-objectif. Alors que s'il y a plusieurs objectifs à optimiser, le problème est dit multi-objectif. Et selon le nombre de variables, il y a le PO à petite échelle et le PO à grande échelle.

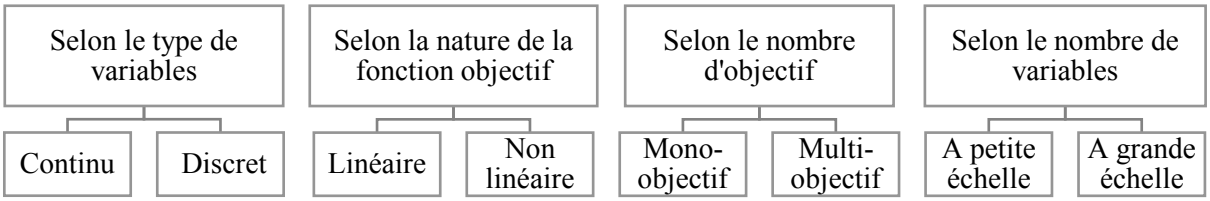


Figure 1.1 : Classification des problèmes d'optimisation

1.3. L'optimisation multi-objectif

1.3.1. Définition d'un POMO

D'un point de vue formel, le problème d'optimisation multi-objectif (POMO), sous sa forme la plus générique, est posé comme suit [11, 15]:

$$minimiser_X \quad F(x) = [F_1(x), F_2(x), \dots, F_k(x)]^T \tag{1.1}$$

Sujet aux contraintes : $g_j(x) \leq 0, j = 1,2,3, \dots, m$

$$h_l(x) = 0, l = 1,2,3, \dots, e$$

$$a_i \leq x_i \leq b_i$$

Où k est le nombre de fonctions « objectif », m est le nombre de contraintes d'inégalité et e est le nombre de contraintes d'égalité. $x \in E^n$ est le vecteur de variables de décision, où n est le nombre de variables x_i. Ces variables sont limitées, parfois, dans une plage de valeurs $[a_i \ b_i]$. $F(x) \in E^k$ est le vecteur de fonctions « objectif » tel que $F_i(x) : E^n \rightarrow E^1$; $F_i(x)$ est appelée aussi objectif, critère, fonction de coût ou fonction de valeur. Les k fonctions objectifs peuvent être linéaires ou non-linéaires, continues ou discrètes, et le vecteur de variables de décision peut également être continu ou discret [16].

L'espace de solutions faisables ou possibles X, appelé aussi espace de décision ou ensemble de contraintes, est défini comme suit [12]:

$$\{x | g_j(x) \leq 0, \ j = 1,2,3, \dots, m \ \ et \ h_l(x) = 0, l = 1,2,3, \dots, e \ et \ a_i \leq x_i \leq b_i\}$$

L'espace des critères faisables ou l'ensemble accessible Z est défini comme $\{F(x) | x \in X\}$. Bien que les termes d'espace de critères faisables et ensemble accessible sont tous deux utilisés dans la littérature pour décrire Z [11], il y a une distinction subtile entre les concepts de faisabilité et d'accessibilité. La faisabilité implique qu'aucune contrainte n'est violée et $F(x)$ appartient à l'ensemble où toutes les contraintes sont vérifiées alors que l'accessibilité implique qu'un point dans l'espace des critères correspond à un point de l'espace de décision [15] (voir la figure 1.2).

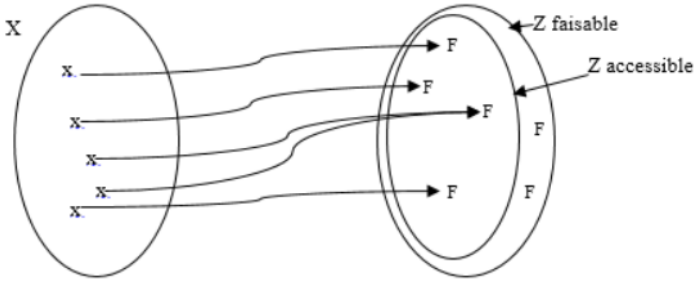


Figure 1.2 : La différence entre faisabilité et accessibilité

1.3.2. Dominance

Contrairement à l'optimisation mono-objectif, l'optimisation multi-objectif est incapable de trouver une solution globale. Elle se base sur le principe de dominance pour rechercher un ensemble de solutions dites points non-dominés. On dit que le vecteur x domine un vecteur y si et seulement si x est au moins aussi bon que y sur tous les objectifs et x est strictement meilleur que y sur au moins un objectif [17]. D'un point de vue formel, ceci s'exprime comme suit :

$$F_i(x) \leq F_i(y) \; \forall \; i \in \{1,2,\dots,k\}$$

$$et \;\; \exists \; j \; \{1,2,\dots,k\} \;\; tel \; que \; F_j(x) < F_j(y)$$

Notons que pour toute paire (x, y) de l'espace de solutions réalisables, une et une seule des affirmations suivantes est vraie [14] : x domine y, x est dominé par y ou x et y sont équivalentes (c-à-d. incomparables) au sens de dominance (voir la figure 1.3).

1.3.3. Optimalité de Pareto et front de Pareto

Un point $x^* \in X$ est dit un optimum de Pareto (ou un point efficace global, non-dominé ou point non-inférieur) pour un POMO si et seulement s'il n'existe pas un $x \in X$ qui le domine [11, 16, 12, 18] (voir la figure 1.3). Toutes les solutions du front de Pareto sont des solutions sur le contour de l'espace des critères faisables Z [11, 12, 13]. En d'autres termes, le front de Pareto ou la frontière Pareto est construite avec les solutions x^* qui dominent toutes les autres mais ne se dominent pas entre elles (voir la figure 1.3).

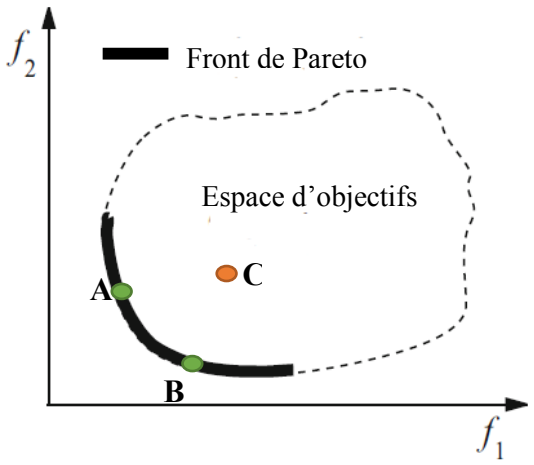


Figure 1.3 : Illustration des notions de dominance, optimalité de Pareto et front de Pareto

Comme nous pouvons le constater sur la figure 1.3, les deux points A et B dominent le point C mais A et B sont incomparables au sens de dominance. Ces deux points sont des optimums globaux de Pareto et appartiennent à l'ensemble des solutions du front de Pareto.

1.3.4. Convexité et concavité

Certaines méthodes de résolution de problèmes d'optimisation multi-objectif nécessitent de respecter certaines hypothèses comme celles de convexité. Comme le montre la figure 1.4, on dit qu'un ensemble X est convexe si en prenant deux points distincts quelconques de cet ensemble, le segment qui les relie est continu dans X, sinon on dit qu'il est concave, formellement $\forall\, x_1, x_2 \in X\ x = \theta\, x_1 + (1-\theta)x_2, x \in X$. Une fonction F est dite convexe sur X si pour chaque paire $(x1, x2) \in X$, $F(\theta\, x_1 + (1\text{-}\theta)x_2) \leq \theta\, F(x_1) + (1\text{-}\theta)F(x_2)$, tel que $0 \leq \theta \leq 1$ [16].

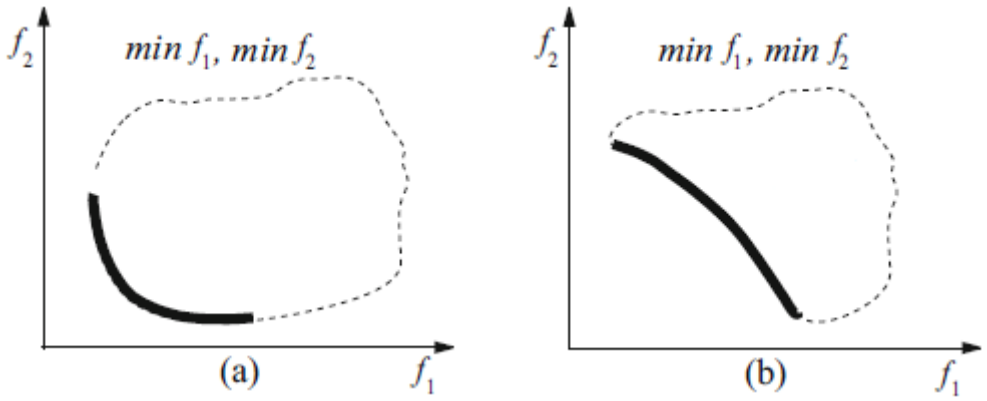


Figure 1.4 **:** Exemples de deux espaces faisables (a) un ensemble convexe et (b) un ensemble concave

1.3.5. Points de référence

Point idéal

Un point $F° \in E^k$, est un point idéal si : $\forall\, i = 1,2,3,\dots,k\,;\, F° = \min\{F_i(x) | x \in X\}$, il correspond à l'optimum de chaque F_i considérée séparément [16]. Ce point ne correspond pas à une solution réalisable, $F° \in (E^k - Z)$ car si c'était le cas, cela sous-entendrait que les objectifs ne sont pas en conflit [14], ce qui ramènerait le problème à un problème ayant une seule solution optimale de Pareto F°. Puisque le point idéal F° est inaccessible, la prochaine meilleure action à faire est de trouver une/plusieurs solution(s) qui soient aussi proche que possible de ce point idéal (voir la figure 1.5).

Point Nadir et Point anti-idéal

Il existe deux approches pour déterminer le point Nadir [11], l'une le définie comme suit : On dit que F^{nad} est un point Nadir, si $\forall\, i = 1,2,3,\dots,k\,;\, F^{nad} = \max\{F_i(x) | x \in X\}$. F^{nad} a comme valeur pour chaque objectif la valeur maximale, la plus mauvaise, des objectifs considérés. Le point Nadir dans ce cas est appelé parfois point anti-idéal ; quoique c'est deux points différents (voir la figure 1.5). Avec la deuxième approche, le point F^{nad} est bien plus difficile à trouver. Dans ce cas, à la différence du point idéal qui représente les bornes inférieures de chaque objectif dans l'espace faisable, le point Nadir correspond à leurs bornes supérieures sur la surface de Pareto et non pas sur tout l'espace faisable [14]. Dans ce cas, le point Nadir sert à restreindre l'espace de recherche, il est utilisé dans certaines méthodes d'optimisation interactives [11, 19] (voir la figure 1.5).

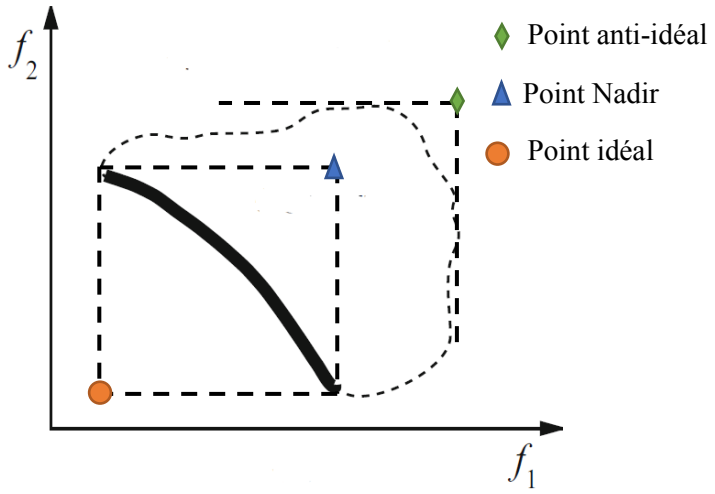


Figure 1.5 : Illustration des points idéal, anti-idéal, nadir et le front de Pareto (FP)

La figure 1.5 illustre les points définis ci-dessus et leur position par rapport au front de Pareto. D'où on peut déduire que le point idéal et le point nadir restreignent l'espace de recherche.

1.3.6. Normalisation

Les deux points, « idéal » et « Nadir » sont utilisés pour la transformation de fonctions objectif, afin que toutes ces fonctions aient la même dimension ; pour toute F_i $0 \leq F_i^{norm} \leq 1$. Plusieurs approches sont proposées à cet effet. Mais, l'approche la plus robuste pour cette transformation qui est, d'ailleurs, considérée comme une normalisation reste la suivante [14, 11]:

$$F_i^{norm} = F_i - F_i^{\circ} / F_i^{nad} - F_i^{\circ} \; pour \; i = 1,2,3, \ldots, k \tag{1.2}$$

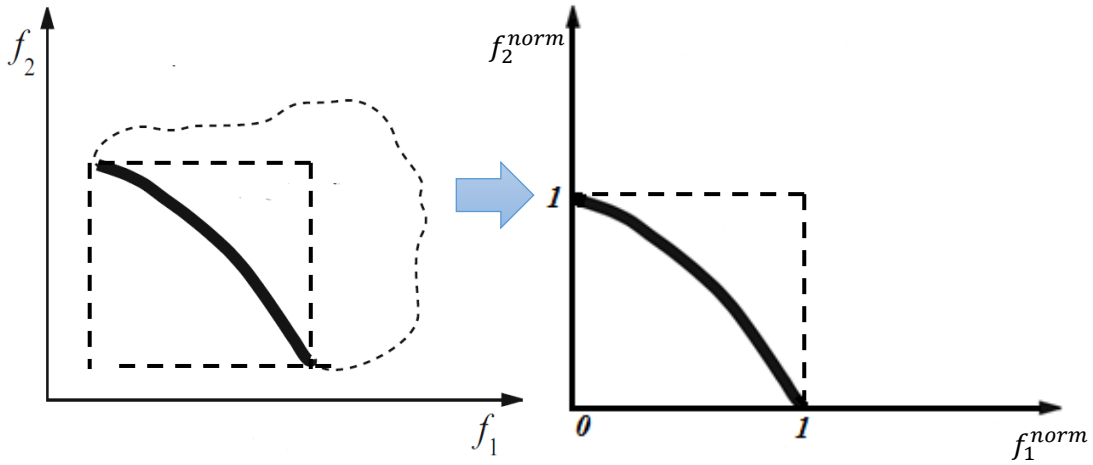


Figure 1.6. Normalisation de l'espace d'objectif

La figure 1.6 illustre un espace d'objectifs avec deux objectifs normalisés, où les valeurs des deux objectifs sont comprises entre 0 et 1. Ce qui permet de traiter ces objectifs d'une manière équivalente dans certaines méthodes, comme la méthode de la somme pondérée par exemple.

1.3.7. Condition nécessaire et condition suffisante

Une formulation fournit une condition nécessaire, permettant de vérifier que chaque point optimum de Pareto est une solution à cette formulation [20]. Cependant, certaines solutions obtenues en utilisant cette formulation peuvent être non optimales au sens de Pareto [21]. D'autre part, si une formulation fournit une condition suffisante, alors sa solution est toujours optimale au sens de Pareto, bien que certains points optimums de Pareto puissent être inaccessibles en utilisant cette formulation [11].

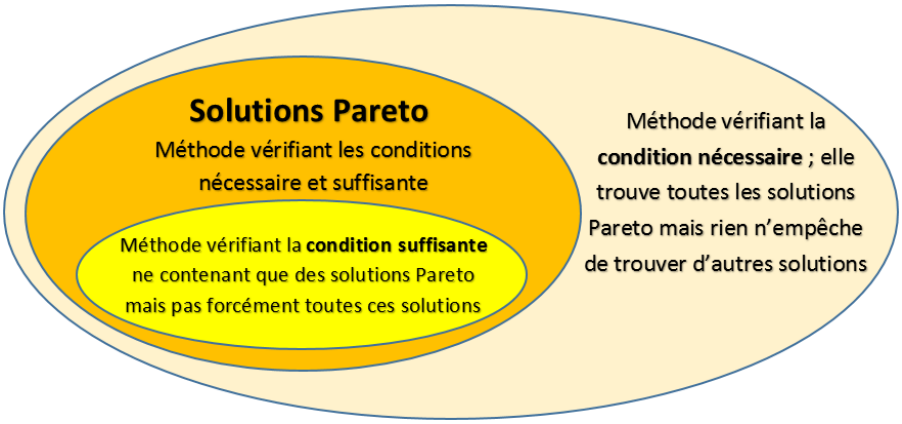


Figure 1.7 : Différence entre les conditions nécessaires, suffisantes et nécessaires & suffisantes.

1.4. Résolution des problèmes d'optimisation multi-objectif

Le processus de résolution d'un problème passe par deux étapes principales : l'optimisation et la prise de décision. Dans la première, on cherche à trouver le(s) meilleur(s) compromis. Et dans la deuxième, le décideur choisit la solution à retenir. Une relation de coopération entre le décideur et le solveur est établie de 4 manières différentes ce qui nous donne 4 approches de résolution, selon le moment d'articulation de préférences de décideur pendant le processus d'optimisation, nommées : sans articulation, avec articulation apriori, avec articulation apostériori et avec articulation interactive, décrites dans la section suivante. Après avoir choisi l'approche, le concepteur choisi l'une des stratégies existantes, classées en : stratégie scalaire, stratégie Pareto et stratégie non-scalaire non-Pareto. Le choix de la stratégie est affecté par le choix de l'approche. Généralement, pour une approche apriori, une stratégie de scalarisation est mise en œuvre. Alors que pour une approche avec articulation apostériori, la stratégie de Pareto est mise en œuvre en utilisant différentes méthodes et algorithmes. Mais, cette règle n'est pas fixe. Par exemple, la scalarisation peut être utilisée avec une approche apostériori, en répétant la scalarisation plusieurs fois avec différents paramètres. Le tableau 1.1 résume la classification des méthodes et algorithmes de résolution d'un POMO selon les différents critères.

Tableau 1.1 : Classification des algorithmes de résolution des POMO

Les approches	Sans articulation	Avec articulation apriori	Avec articulation interactive	Avec articulation apostériori
Les stratégies	Scalaire	Non-scalaire non-Pareto	Pareto	
Les méthodes et les algorithmes	Programmation mathématique		Métaheuristiques	
	WS, NBI, NC, …		SA, GA, PSO, TLBO, …	

1.5. Les approches de résolution

La manière dont un POMO peut être résolu du point de vue décideur est classée en quatre approches (voir la figure 1.8) :

1.5.1. Avec articulation apriori de préférences

Cette approche permet au décideur de spécifier ses préférences, articulées en termes de buts et d'importance accordée à chacun de ces buts, avant même de lancer la procédure d'optimisation. La plupart des algorithmes suivent la stratégie de scalarisation en intégrant des paramètres sous forme de coefficients, d'exposants ou des limites de contraintes. Cette approche est simple, mais la difficulté c'est que le décideur ne connait pas nécessairement à l'avance les possibilités et les limites du problème ; de ce fait, il peut avoir des aspirations trop optimistes ou trop pessimistes.

1.5.2. Avec articulation apostériori de préférences

Parfois, il est difficile pour le décideur de donner une approximation explicite de ses préférences avant même d'entamer le processus d'optimisation. Dans ce cas, il est plus judicieux de retarder la décision et de chercher en premier un ensemble de solutions optimales candidates au sens de Pareto. Cette approche donne au décideur un aperçu sur les différentes solutions disponibles, mais s'il y a plus de deux objectifs dans le problème, il devient difficile pour le décideur d'analyser la grande quantité d'informations qui en découle ; la visualisation de solutions n'est plus aussi simple que dans le cas bi-objectif. En plus, la génération de l'ensemble des solutions optimales de Pareto peut être coûteuse en calcul. L'avantage c'est que les solutions générées par ces méthodes sont indépendantes des opinions du décideur et restent valables même si les opinions sont changées, on n'est pas obligé de relancer le processus de résolution ; il suffit de sélectionner une autre solution.

1.5.3. Avec articulation interactive de préférences

Un algorithme de résolution itératif est proposé. Après chaque itération, certaines informations sont fournies au décideur afin qu'il puisse spécifier des informations de préférence. Ce processus est ensuite répété jusqu'à ce que le décideur soit satisfait du résultat obtenu. De cette manière, le décideur dirige le processus de résolution vers sa zone préférée [19]. Ce qui conduit à la génération et l'évaluation d'une seule partie de solutions optimales. Le décideur n'a pas besoin d'avoir une structure globale de préférence et il peut apprendre pendant le processus de résolution. Ceci est un avantage très important des méthodes interactives, car apprendre à connaitre le problème, ses possibilités et ses limites est souvent très précieux pour le décideur. Les méthodes interactives surmontent les faiblesses des méthodes à priori et a posteriori parce que le décideur n'a pas besoin d'une structure globale de préférence et que les solutions Pareto optimales générées sont guidées par les intérêts du décideur à chaque étape (solution ad hoc). Ceci signifie des économies en coût de calcul. En plus, ceci évite la nécessité de comparer de nombreuses solutions Pareto optimales simultanément.

1.5.4. Sans articulation de préférences

Les méthodes de cette approche ne nécessitent aucune information sur les préférences du décideur. Elles exigent que le décideur puisse accepter la solution obtenue sans participer dans le processus de recherche. La plupart des méthodes de cette approche sont des simplifications des méthodes avec articulation apriori. Ces méthodes sont utilisées dans le cas où toutes les solutions sont équivalentes et indifférentes pour le décideur. Mais dans la réalité, il est rare de rencontrer ce type de problèmes.

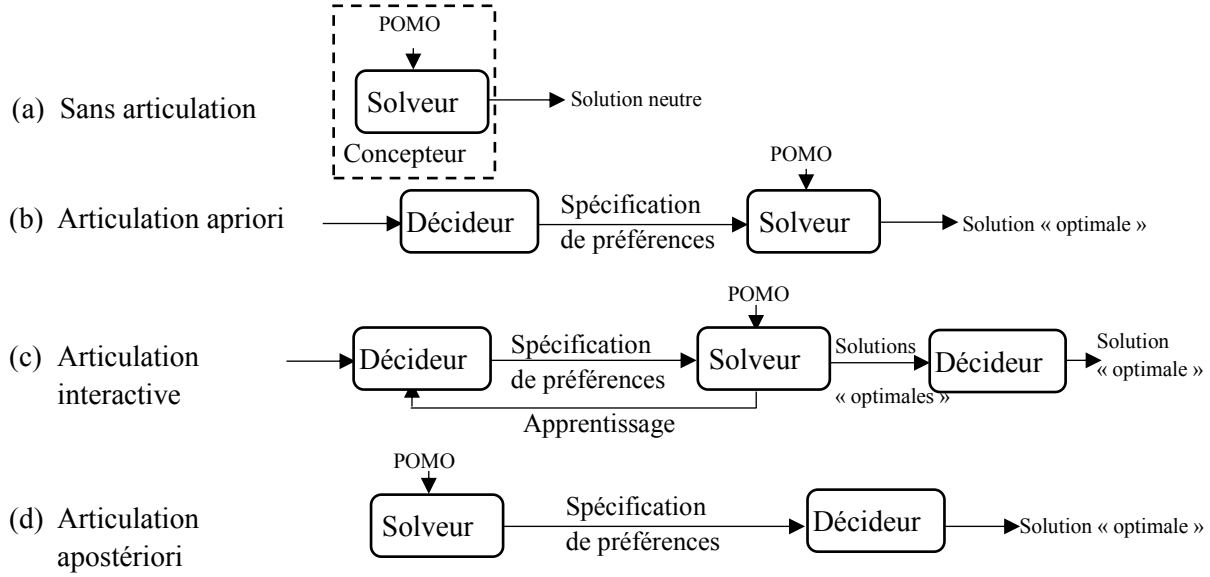


Figure 1.8 : Processus de résolution d'un POMO selon l'approche

1.6. Les stratégies de résolution

Du point de vue du concepteur, les méthodes et les algorithmes de résolution d'un POMO suivent trois stratégies, décrites dans les sous-sections suivantes.

1.6.1. Scalaire

Le problème multi-objectif sera transformé dans ce cas en un problème mono-objectif. Parmi les méthodes qui suivent cette stratégie : la méthode de la somme pondérée (WS), ε-contrainte, programmation par but. Les méthodes appartenant à cette classe sont généralement des méthodes de programmation mathématique avec une articulation apriori de préférences. Elles sont simples mais sensibles à la forme du front de Pareto.

A. La somme pondérée

La méthode de la somme pondérée transforme le POMO en [12] :

$$U(x) = \sum_{i=1}^{k} w_i F_i(x) \tag{1.3}$$

La minimisation de l'équation (1.3) est suffisante pour l'optimalité de Pareto mais elle ne fournit pas des solutions dans une concavité, donc elle ne fournit pas la condition nécessaire [11] [21]. Kim et De Weck [22] ont proposé la méthode de la somme pondérée adaptative qui est capable de trouver des points Pareto dans des régions concaves.

B. Programmation par but

Charnes et al. [23] ont développé une méthode dite : « programmation par but », en associant à chaque fonction $F_j(x)$ un but b_j , ensuite la déviation totale $\sum_{j=1}^{k}|d_j|$ est minimisée, où d_j correspond à la déviation de $F_j(x)$ par rapport à b_j. Pour modéliser la valeur absolue présentée ci-devant, d_j est divisée en deux parties telles que : $d_j = d_j^+ - d_j^-$, avec $d_j^- \geq 0$ et $d_j^+ \geq 0$, $d_j^+ d_j^- = 0$ (d_j^+ et d_j^- sont exclusives) ; par conséquent, $|d_j| = d_j^+ + d_j^-$.

d_j^+ et d_j^- représentent, respectivement, la sous-réalisation et la sur-réalisation du but b_j. Le problème d'optimisation est reformulé, alors, comme suit [11] :

$$\min_{x \in X, d^-, d^+} \sum_{i=1}^{k} (d_i^- + d_i^+) \tag{1.4}$$

$$\text{Sujet à : } F_j(x) + \left(d_j^+ - d_j^-\right) = b_j\,, j = 1,2 \ldots, k;$$

$$d_j^-, d_j^+ \geq 0,, j = 1,2, \ldots, k;$$

$$d_j^+ d_j^- = 0,, j = 1,2, \ldots, k.$$

Cependant, en dépit de sa popularité et sa large gamme d'applications, il n'y a aucune garantie qu'elle offre une solution optimale au sens de Pareto à cause de variables supplémentaires et des contraintes d'égalité non linéaires, qui peuvent être gênantes avec des problèmes plus importants.

C. Fonction objectif bornée

Cette méthode minimise une seule fonction objectif, celle qui est la plus importante et qu'elle note $F_s(x)$, toute autre fonction objectif se transforme en une contrainte telle que :

$$l_i \leq F_i(x) \leq \varepsilon_i\,, i = 1,2, \ldots, k\,; i \neq s \tag{1.5}$$

où l_i *et* ε_i sont les bornes de la fonction objectif $F_i(x)$. Haimes et al. [24] ont introduit l'approche $\varepsilon - contrainte$ (appelée également e-contrainte ou approche de compromis), dans laquelle l_i est exclue. Une variante de l'approche $\varepsilon - contrainte$ a été proposée dans [25], où Fu et Diwekar ont prouvé que leur algorithme est capable de trouver toutes les solutions de Pareto même sous la contrainte de la concavité.

1.6.2. Non-scalaire et non-Pareto

Les méthodes qui suivent cette stratégie ne transforment pas le problème en un problème mono-objectif et n'utilisent pas la notion de dominance de Pareto ; les objectifs sont traités séparément. On présente une méthode mathématique et une autre méta-heuristique : la première nécessite une connaissance apriori pour arranger les objectifs, tandis que la deuxième est classée comme méthode apostériori. L'aspect de l'optimisation multi-objectif est contourné dans les méthodes suivant cette stratégie. Les solutions générées sont largement optimisées pour un seul objectif.

A. La sélection lexicographique

Dans cette méthode lexicographique, les objectifs sont arrangés par ordre de leur importance. Ensuite, le problème de l'OMO est résolu, en traitant un objectif à la fois ; la fonction la plus préférée F_1 est calculée sans contrainte :

$$\begin{aligned} &\text{Minimiser}_{\,x\epsilon X} \quad F_1(x) \text{ sans contrainte} \\ &Minimiser_{\,x\epsilon X} \quad F_i(x)\ \ i = 2,3, \ldots, k \\ &sujet\ \text{à} \qquad\quad F_j(x) \leq F_j\left(x_j^*\right), j = 1,2, \ldots, i-1 \end{aligned} \tag{1.6}$$

Dans cette équation, i représente la position de la fonction dans la séquence de préférence et $F_j\left(x_j^*\right)$ représente l'optimum de la fonction F_j , trouvé dans la j[ième] itération et ajouté aux j-1 contraintes pour avoir j contraintes dans la j[ième]+1 itération. « Compendious lexicographic method », une version modifiée de la méthode lexicographique suggérée dans [26], est spécifiée pour les problèmes multi-objectif linéaires dans les cas de deux et trois variables de décision.

B. La sélection parallèle

L'exemple de cette classe est l'algorithme génétique VEGA (Vector Evaluated Genetic Algorithm). Dans VEGA, la population de taille *N* est divisée en *k* sous-populations tel que *k* est le nombre

d'objectifs. Les objectifs sont traités indépendamment les uns des autres et les individus d'une sous-population sont évalués selon leur objectif. À la fin de chaque génération, les individus sont mélangés et regroupés aléatoirement. L'évaluation du même individu change d'une génération à une autre selon l'objectif de la sous-population. Il a été prouvé que son comportement est similaire à la méthode d'agrégation.

1.6.3. Pareto

Dans cette stratégie, la notion de dominance est l'axe principal ; où des objectifs contradictoires peuvent être optimisés simultanément. Les algorithmes suivant la stratégie de Pareto sont généralement des méta-heuristiques utilisées dans le cas où un ensemble de solutions soit requis et les préférences du décideur sont articulées d'une manière apostériori. Les algorithmes doivent prendre en compte deux objectifs principaux : L'intensification et la diversification [27] (voir la figure 1.9).

- **L'intensification** : Afin de converger vers la frontière de Pareto, plusieurs mécanismes sont développés et classés en : mécanismes de classement (ranking), mécanismes d'élitisme.
- **La diversification** : Et afin de trouver des solutions bien distribuées au long de la frontière de Pareto, des mécanismes de diversification sont utilisés.

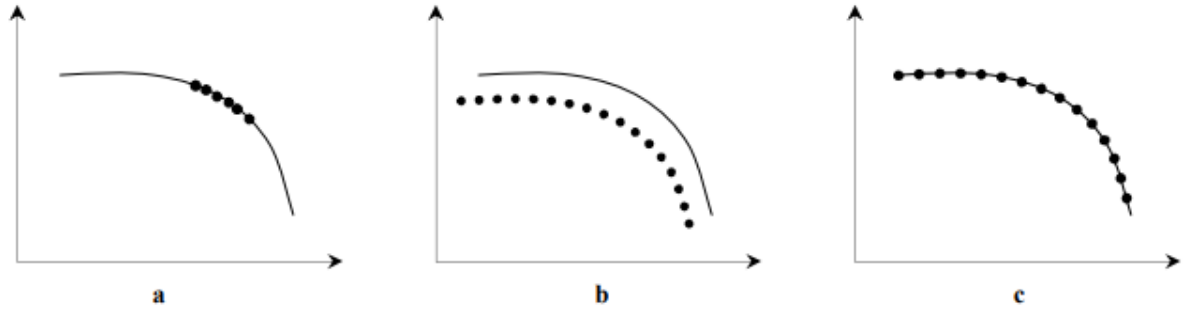


Figure 1.9 : L'intensification et la diversification

Dans la figure 1.9, (a) présente une bonne intensification et une mauvaise diversification, (b) présente une bonne diversification et une mauvaise intensification, (c) présente le bon équilibre entre les deux objectifs.

A. Les mécanismes du ranking

À cause de l'existence de plusieurs objectifs à optimiser, il n'existe pas une relation d'ordre total entre les solutions. En utilisant la notion de dominance, les solutions obtenues dans une génération (itération) sont classées suivant différents mécanismes développés. Fonseca et Fleming ont proposé et utilisé le mécanisme de ranking basé sur la notion de dominance de Pareto, dans l'algorithme génétique MOGA pour la première fois [28]. À la génération t, on donne un rang pour chaque individu x_i, égal au nombre $p_i(t)$d'individus qui le dominent plus 1, $rank\ (x_i, t) = 1 + p_i(t)$. Par la suite, les individus sont classés selon leur rang. Après, une fonction d'efficacité F est affectée pour chaque individu, calculée en fonction du rang : $F = 1 / rank\ (x_i, t)$.

Tandis que, le mécanisme du tri non-dominé (NDS : Non-Dominated Sorting), proposé par Goldberg [29], a été utilisé dans l'algorithme NSGA [30](Non-dominated Sorting Genetic algorithm). Le principe de fonctionnement de ce mécanisme est comme suit. Pour assigner un rang à chaque individu, le nombre d'individus qui le dominent est calculé, ainsi que les individus qu'il domine sont trouvés. Comme le montre la figure 1.10, le rang 1 est assigné aux individus qui ne sont pas dominés du tout. Le rang 2 est assigné aux individus dominés, seulement, par ceux du rang 1. Et les individus du rang 3 sont dominés par ceux du rang 1 et 2, …etc.

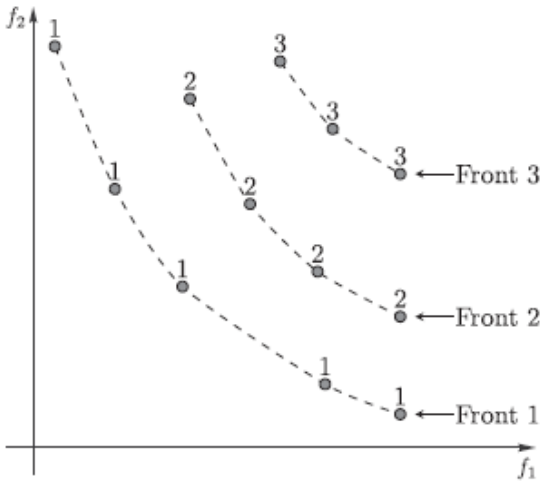


Figure 1.10 : Mécanisme de ranking NDS [31]

B. L'élitisme

On peut distinguer deux types d'élitisme : (1) en utilisant une archive externe, les meilleurs individus non-dominés découverts à partir de la première génération jusqu'à la génération actuelle sont archivés dans une population externe. Cette population est continuellement mise à jour. Les individus de cet ensemble participent dans la génération d'une nouvelle population avec une certaine probabilité. (2) ou bien en utilisant la population elle-même. La nouvelle population générée est mélangée avec la population actuelle. Par la suite, les meilleurs individus des deux populations sont sélectionnés pour construire la population de la génération suivante. L'exemple de ce type est NSGA-II [32]. Il a été prouvé que l'élitisme augmente la vitesse de convergence vers le front de Pareto [32].

C. Les mécanismes de diversification

Afin de fournir au décideur un ensemble de solutions représentatif, des mécanismes sont développés pour maintenir la diversité tels que : nichage séquentiel [33], restriction de voisinage [34], la distance d'encombrement (Crowding distance CD) [32], fonction de partage (Sharing function) [35].

La fonction de partage, utilisée pour la première fois dans l'algorithme NSGA [30], permettant de dégrader la fonction d'efficacité F d'un individu par rapport au nombre d'individus semblables ; plus les individus sont regroupés, plus leur valeur d'efficacité est faible. On utilise un compteur de niche donnant une estimation du nombre d'individus qui se trouvent dans le voisinage de chaque individu i. On peut distinguer deux types de niches : génotypiques et phénotypiques. Une niche phénotypique permet de régler la diversité des solutions dans l'espace des fonctions objectif alors que la niche génotypique permet de régler la diversité des solutions dans l'espace de décision. La niche combinée des deux permet d'obtenir des solutions diversifiées aussi bien dans l'espace de décision que dans l'espace des fonctions objectif [36]. Les bornes de voisinage (niche) sont déterminées par la distance maximale prédéfinie $\sigma share$. La distance entre l'individu i et tout individu j dans son voisinage est calculée. Le plus cette distance $d(i,j)$ est proche de $\sigma share$, la valeur $Sh\ (i,j)$dégrade. Par la suite, la valeur de la fonction de partage de l'individu i est calculée par la somme de $Sh\ (i,j)$ des individus dans son voisinage. Les meilleurs individus sont ceux ayant un petit rang et moins d'individus dans son voisinage et une $Sh\ (i,j)$ maximale. L'inconvénient majeur de la fonction de partage est la difficulté de réglage de la valeur $\sigma share$.

Pour cette raison, dans la version améliorée NSGA-II [32], Deb et al. ont utilisé un autre mécanisme appelé distance d'encombrement (voir la figure 1.11). Pour chaque fonction objectif k, les individus du même front sont triés par ordre croissant selon la valeur de cette fonction objectif. Les deux individus 1 et l à l'extrémité de la liste d'individus triée L ont un $Crowddist_k^1 = \infty, Crowddist_k^l = \infty$. Pour chaque individu i parmi les individus classés entre 2 et $l-1$:

$$Crowddist_k^i = \frac{F_k^{i+1} - F_k^{i-1}}{F_k^l - F_k^1} \qquad (1.7)$$

La distance d'encombrement d'un individu i est calculée par la somme des distances d'encombrement (équation 1.8) calculées pour chaque objectif en utilisant l'équation (1.7). L'utilisation de la somme nécessite la normalisation des fonctions objectifs.

$$Crowddist^i = \sum_{k=1}^{m} Crowddist_k^i \qquad (1.8)$$

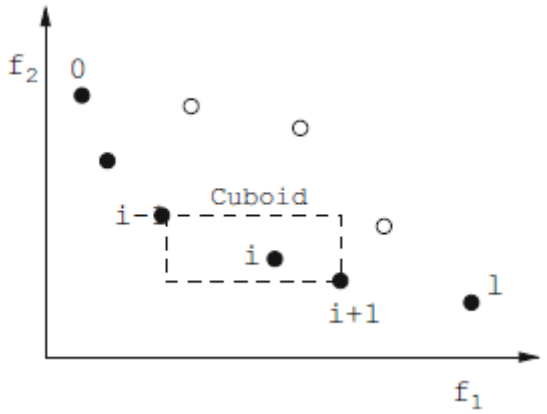


Figure 1.11 : Calcul de la distance d'encombrement

1.7. Les méthodes de résolution

Les méthodes présentées dans cette section peuvent être classées sous différentes stratégies et approches. Comme le montre la figure 1.12, ces méthodes peuvent être des méta-heuristiques ou bien des méthodes de programmation mathématique. Ces dernières peuvent être des algorithmes **exacts** ou approchés. Les algorithmes **exacts,** tels que le branch and bound multi-objectif [37] et la programmation dynamique multi-objectif [38], fournissent le front de Pareto réel mais ils ne sont efficaces que pour des problèmes de petite taille à deux objectifs seulement. Les méthodes mathématiques **approchées** telles que : la somme pondérée, la somme pondérée adaptative [22], la méthode lexicographique compendious [26], la programmation par but [23], la programmation physique [39], la méthode d'intersection de la bordure et la normale (NBI : Normal boundary Intersection) [40, 41], la méthode de contrainte normale (NC : Normal Constraint) [1]et d'autres (voir [11]).

Les méta-heuristiques sont des algorithmes généraux dédiés aux problèmes d'optimisation difficiles telle que l'optimisation d'un problème avec 3 objectifs ou plus, avec un espace de recherche vaste ou bien un problème discret. Les méta-heuristiques sont des méthodes approchées qui donnent une approximation du front de Pareto mais pas le front de Pareto réel. On distingue deux types de méta-heuristiques : les algorithmes basés sur une solution unique et les algorithmes basés sur une population. Les algorithmes appartenant à la première classe travaillent sur un seul point de l'espace de décision. Le processus de recherche commence par un point aléatoire ensuite une amélioration itérative est garantie par la sélection d'une solution meilleure dans le voisinage de la solution actuelle. Tandis que la

deuxième classe repose sur l'utilisation d'une population d'individus. L'intérêt de ce type d'algorithmes est leur capacité à explorer une vaste partie de l'espace en une seule itération.

La première classe d'algorithmes basés sur une solution unique regroupe une variante d'algorithmes tels que la méthode recuit simulé multi-objectif (MOSA : Multi-Objective Simulated Annealing) [42, 43] inspiré du processus de recuit en métallurgie afin d'équilibrer l'énergie des métaux, l'algorithme de la recherche tabou multi-objectif (MOTS : Multi-Objective Tabu Search algorithm) [44] a pris son nom de la liste taboue utilisée pour interdire le retour vers des solutions déjà explorées, l'algorithme GRASP (Greedy Randomized Adaptative Search procedure) [45], l'algorithme de recherche par variable de voisinage (MO-VNS : MO Variable Neighborhood Search) [46], la recherche locale guidée (MO-GLS : MO Guided Local Search) [47].

Tandis que la deuxième classe d'algorithmes basés sur la population est plus riche d'algorithmes et plus adaptée dans la littérature pour obtenir une approximation du front de Pareto. Les algorithmes de cette classe sont discutés dans la section suivante.

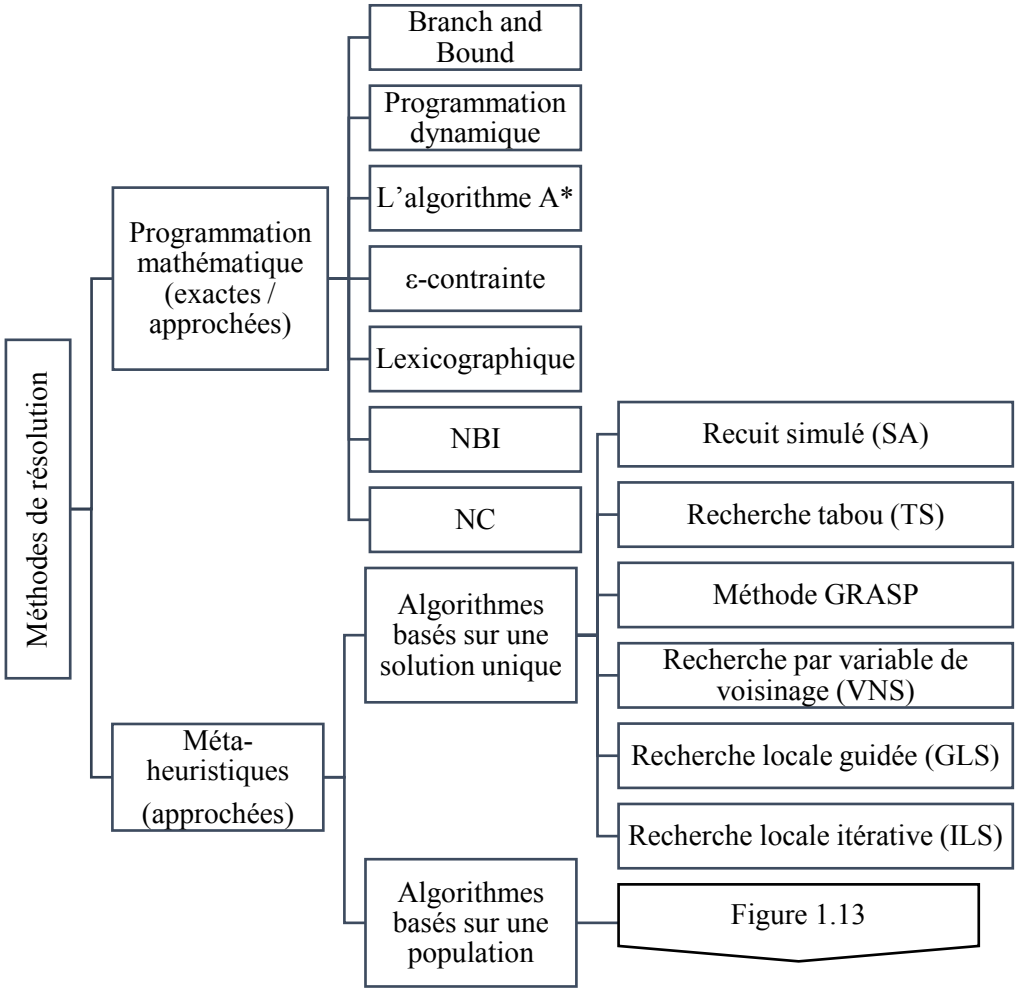


Figure 1.12 : Classification des méthodes de résolution de POMO du point de vue du concepteur

1.8. Les algorithmes basés sur la population

Une variété d'algorithmes basés sur la population sont généreusement adaptés aux POMOs. La littérature les a classés en plusieurs catégories en fonction de la source d'inspiration. La figure 1.13 présente les catégories les plus connues où les algorithmes en gris sont classés comme des algorithmes de l'ancienne génération et le reste sont classés comme des algorithmes de la nouvelle génération [48].

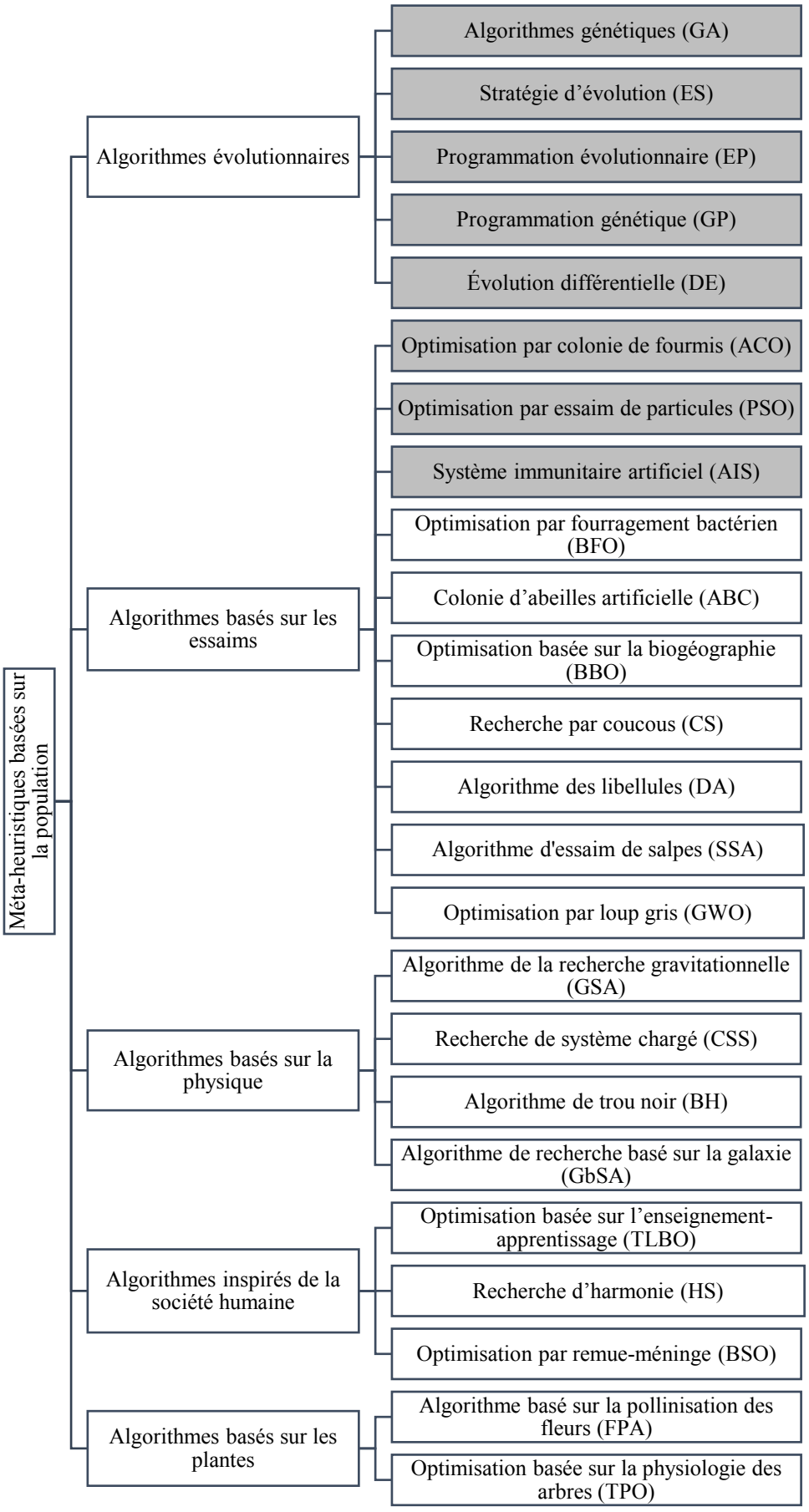


Figure 1.13 : Taxonomie des méta-heuristiques basées sur la population

1.8.1. Les algorithmes évolutionnaires

Les algorithmes évolutionnaires font partie des premiers algorithmes inspirés de la biologie, utilisés pour résoudre les problèmes d'optimisation en général et les problèmes multi-objectif en particulier, les algorithmes évolutionnaires [49]. Ces algorithmes sont inspirés de mécanismes d'évolution biologique, qui portent sur le développement des générations successives contenants un ensemble d'individus appelé population [50]. Les algorithmes évolutionnaires reposent sur 3 éléments principaux : la population, la fonction de fitness et le mécanisme d'évolution. Les individus de la population sont caractérisés par un chromosome représentant la solution dans l'espace de décision. La valeur de la fitness mesure sa qualité dans l'espace objectif. Alors que le mécanisme d'évolution permet d'éliminer les individus ayant une fitness faible en utilisant l'opérateur de sélection et générer de nouveaux individus en utilisant les deux opérateurs de mutation et croisement.

L'algorithme génétique élitiste (NSGA-II : Non-dominated Sorting Genetic Algorithm) [32] garantit l'élitisme en utilisant la population principale, sans archive externe. Le mécanisme du tri NDS et le mécanisme de la distance d'encombrement sont utilisés pour garantir l'intensification et la diversification. Tandis que, la stratégie d'évolution assistée par des substituts (SMES : Surrogate-assisted Multi-objective Evolution Strategy) [51] est basée sur la substitution pour prédire les valeurs des fonctions objectifs ce qui élimine le processus de la simulation pour obtenir ces derniers. Et plusieurs autres algorithmes tels que la programmation évolutionnaire (MOEP : Multi-Objective Evolutionary Programming) [52], la programmation génétique (MOGP : Multi-Objective Genetic Programming) [53] et l'évolution différentielle (MODE : Multi-Objective Differential Evolution) [54].

1.8.2. Les algorithmes basés sur les essaims

La deuxième catégorie comprend les algorithmes inspirés des essaims naturels. Comme le présente la figure 1.13 et différemment à la première catégorie qui ne contient que les algorithmes de l'ancienne génération, celle-là contient des algorithmes des deux générations. Parmi les algorithmes de l'ancienne génération : l'optimisation multi-objectif par essaim de particules (MOPSO : Multi-Objective Particle Swarm Optimization) [55] mettant à jour la position d'un ensemble de particules représentées par des vecteurs dans l'espace de décision. L'optimisation par colonie de fourmis (MOACO : MO Ant Colony Optimization) [56] inspirée par le comportement de fourragement observé chez les fourmis. Et l'optimisation par système immunitaire artificiel (MOAIS : MO Artificial Immune System) [57] émulant la réaction protectrice du système immunitaire humain contre les intrus.

La nouvelle génération aussi regroupe une variété d'algorithmes tels que : l'optimisation par fourragement bactérien (MBFO : Multi-objective Bacterial foraging Optimization) [58] inspirée par le comportement de fourragement chez les bactéries. L'optimisation basée sur la colonie d'abeilles artificielle (MOABC : MO Artificial Bee Colony) [59] inspirée par le comportement de recherche d'une source de nourriture chez les abeilles. L'optimisation basée sur la biogéographie (MO-BBO : MO Biogeography Based Optimization) [60] inspirée par le processus de migration d'une ile ou d'un habitat vers un autre et l'apparition et l'extinction des espèces. L'algorithme multi-objectif des libellules (MODA : Multi-Objective Dragonfly Algorithm) [61] inspiré par l'interaction sociale des libellules lors la navigation et la recherche de nourriture. La recherche par coucous (MOCS : MO cuckoo search) [62] émulant la reproduction agressive de certaines espèces de coucous qui pondent leurs œufs dans les nids des autres espèces. L'optimisation par loup gris (MO-GWO : MO gray Wolf Optimization) inspirée de la hiérarchie dominante et stricte observée chez les loups gris [63]. L'algorithme de l'essaim de salpes (MSSA: Multi-objective Salp Swarm Algorithm) inspiré par le comportement de recherche de nourriture des salpes dans l'océan [64].

1.8.3. Les algorithmes basés sur la physique

La troisième catégorie regroupe les algorithmes basés sur la physique, par exemple l'algorithme de la recherche gravitationnelle (MOGSA : MO Gravitational Search Algorithm) [65] émulant les forces gravitationnelles entre les particules. L'algorithme de la recherche basée sur système chargé (MOCSS : MO Charged System Search) [66] où les individus sont considérés comme des particules chargées qui peuvent appliquer des forces électriques les uns sur les autres ce qui résulte des mouvements de particules électriques avec une certaine vitesse et accélération. L'optimisation basée trou noir (MOBH : MO Black Hole) [67] dont le principe est de guider les individus vers les meilleurs individus de l'itération actuelle, appelé trous noirs. Par la suite, les individus dans les champs des trous noirs seront remplacés par de nouveaux individus. L'algorithme de recherche basé sur la galaxie (GbSA : Galaxy based Search Algoorithm) [68] imite le mouvement en spirale des galaxies pour fouiller son environnement en échappant les optimums locaux par le chaos (l'aléatoire).

1.8.4. Les algorithmes inspirés de la société humaine

Une catégorie relativement nouvelle comprend les algorithmes basés sur la société humaine. Par exemple, l'algorithme d'optimisation par remue-méninge basé sur la décomposition (MOBSO/D : Decomposition-based Multi-Objective Brain Storm Optimization) [69] simule le brainstorming qui est un comportement collectif observé chez les êtres humains. C'est un algorithme simple basé sur le principe de répartition de solutions en plusieurs clusters. Les nouvelles solutions sont générées en appliquant des opérateurs génétiques sur un ou plusieurs individus. L'algorithme de la recherche d'harmonie (MOHS : MO Harmony Search) [70] imitant l'improvisation des musiciens. L'algorithme utilise une recherche stochastique, une mémoire d'harmonie et un taux d'ajustement de la tonalité pour trouver l'état parfait de l'harmonie et trouver par la suite les solutions optimales. L'optimisation basée sur l'enseignement-apprentissage (MOTLBO : Multi-objective Teaching-Learning Based Optimization) [71] inspirée du processus d'apprentissage traditionnel. Le principe de fonctionnement de cette dernière approche et les améliorations faites sur ses versions sont discutés en détail dans le prochain chapitre.

1.8.5. Les algorithmes basés sur les plantes

D'autres catégories apparaissent encore, comme la catégorie des algorithmes basés sur les plantes [72], tels que l'algorithme de pollinisation des fleurs (MOFPA : Multi-Objective Flower Pollination Algorithm) [73]. L'algorithme d'optimisation basé sur la physiologie des arbres (TPO : Tree Physiology Optimization) [74] inspiré du système de développement des plantes. L'algorithme est basé essentiellement sur le rapport racine-pousse qui définit la régulation de développement et l'équilibre de fonctionnel entre les deux parties supérieure et inférieure. Et d'autres, comme l'optimisation basée sur les herbes envahissantes (MOIWO : MO Invasive Weed Optimization) [75] inspirée de l'écologie et la biologie des mauvaises herbes.

1.9. Domaines d'application de l'optimisation multi-objectif

L'optimisation multi-objectif est appliquée dans une large panoplie de domaines tels que :

Data Mining : Les problèmes les plus connus du Data Mining sont la sélection, la classification, la régression, le regroupement, l'association de règles et la détection de déviation. Mukhopadhyay [76] et [77] ont discuté les différentes approches évolutionnaires proposées pour quatre grands problèmes : la sélection, la classification, le regroupement et l'association de règles. Le choix des algorithmes évolutionnaires est dû à la taille importante des problèmes appartenant au Data Mining.

Ressources d'eau et la conception des réseaux de distribution d'eau : Plusieurs critères doivent être optimisés lors de la conception d'un système de distribution d'eau tel que le coût de réalisation, la

durée de vie, le coût de maintenance, la fiabilité du système et la qualité de l'eau [78], tous ces critères sont souvent contradictoires. Parmi les problèmes de test de réseaux de distribution d'eau : le problème des tunnels de New York (NYT) et le réseau de Hanoi. Et parmi les problèmes de ressources d'eau multi-objectif : HBV, un modèle de Précipitations-Ruissellement qui simule la transformation de précipitations, comme entrées, en ruissellement avec 14 paramètres de contrôle et 4 objectifs ; le problème de surveillance des eaux souterraines (LTM) avec 4 objectifs et un espace de décision discret mais très large [79].

Conception en ingénierie : Nous pouvons citer les deux exemples suivants comme exemples de problèmes multi-objectif de la conception en ingénierie : Le problème de distribution de l'énergie électrique respectueuse de l'environnement et économiquement réalisable dont l'objectif est de sélectionner les sorties de l'unité génératrice qui répondent à la demande d'un coût d'exploitation minimal et causent une pollution minimale, tout en satisfaisant les contraintes de l'unité et du système [80]. Et le problème de conception préliminaire d'un vraquier avec certains critères techniques et économiques [81].

Réseaux informatiques : Le transfert de l'information, dans un réseau informatique, est l'un des problèmes multi-objectif où on doit choisir le chemin qui répond au besoin de minimiser le nombre de sauts, le délai de transmission et la consommation de la bande passante [82]. Dans le cas d'un réseau optique d'autres critères viennent se rajouter à ceux-là, comme la minimisation du nombre de λ et l'atténuation optique [82].

Réseaux sans fil : Le déploiement de ce type de réseaux présente un problème multi-objectif en exigeant la maximisation de la couverture, la minimisation du coût et la minimisation du chevauchement dans le réseau. Gamal et al. [83] ont discuté le problème d'installation d'un ensemble d'émetteurs dans des sites candidats avec trois objectifs. En plus du problème de déploiement, il existe d'autres problèmes dans ce type de réseaux tels que l'ordonnancement et le routage.

Réseaux de capteurs sans fil : Comme dans les réseaux sans fil, les problèmes majeurs sont le déploiement, le routage et l'ordonnancement avec plusieurs objectifs à satisfaire tels que la minimisation d'énergie consommée, minimisation de la densité du réseau, minimisation de la latence, maximisation de la durée de vie avec certaines contraintes telles que le type de capteurs (homogènes/ hétérogènes) et la topologie [84]. Plusieurs travaux sont faits dans ce contexte tel que [85, 86].

Cloud Computing : L'affectation des machines virtuelles sur des machines physiques nécessite la recherche d'une solution optimale parmi les solutions non-dominées qui minimisent simultanément le gaspillage total des ressources et la consommation d'énergie ; ce problème est étudié dans [87]. Mezmaz et al. [88] ont discuté le problème d'ordonnancement d'applications parallèles à contrainte de priorité sur des systèmes de calcul hétérogènes avec deux objectifs : la minimisation du temps d'achèvement et la consommation d'énergie.

Énergie renouvelable : Plusieurs problèmes multi-objectif interviennent dans ce domaine tel que le placement, le dimensionnement, la conception, la planification et le contrôle de ce type de systèmes d'énergie renouvelable et durable [89].

Conception de médicaments : La découverte de médicaments est un problème multi-objectif avec plusieurs critères pharmaceutiques importants et des espaces de solutions vastes et complexes [90].

Robotique : La plupart des tâches robotiques cachent plus qu'un objectif. L'une des techniques utilisées est l'apprentissage par transfert dont le principe est de réutiliser l'expérience acquise de l'apprentissage de satisfaction d'un objectif pour apprendre un nouveau. García et al. [91] ont décrit une

approche en deux étapes. Durant la première étape, l'algorithme apprendra une politique optimisant un objectif principal, en deuxième lieu on essaie d'éviter la dégradation de la performance du système, toute en cherchant une nouvelle politique qui satisfait également un sous-objectif. D'autres problèmes sont traités tels que la planification du mouvement du robot dans un environnement incertain avec dangers [92]. Herrera Ortiz et al. [93] ont proposé un algorithme pour résoudre le problème de navigation de robot autonome avec trois objectifs : minimiser la distance au but, maximiser la distance entre le robot et l'obstacle le plus proche et maximiser la distance parcourue sur chaque configuration de champ.

Aéronautique et aérospatial : une conception optimale en ingénierie aéronautique et aérospatiale est, par nature, un problème multi-objectif et multidisciplinaire. On prend l'exemple de la conception d'un petit avion supersonique en tenant compte de sept objectifs : le poids, la portée, la longueur du champ équilibré au décollage, la sonorité, la surpression, le nombre de Mach de vol et la taille de la cabine ; certains d'entre eux ont été minimisés tandis que d'autres ont été maximisés [94]. Parmi les travaux dans ce contexte, on cite [95, 96].

D'autres domaines sont présentés par Stadler [97] dans son livre, tels que l'économie, la planification des ressources, la gestion des ressources renouvelables et les systèmes de contrôle des avions.

1.10. Conclusion

Ce chapitre passe en revue l'optimisation multi-objectif et les notions connexes les plus importantes pour comprendre ce vaste domaine. Une variété de méthodes sont proposées pour résoudre différents POMO mais il s'agit généralement des problèmes de test continu et de petite taille. Les problèmes réels peuvent présenter des difficultés plus sérieuses que les fonctions benchmark pour trouver des solutions adéquates [98] parlant surtout de problèmes discrets et/ou à grande échelle qui sont plus difficiles à résoudre. Malgré le manque de preuves théoriques qui démontre leur convergence, les méta-heuristiques à base de population ont montré leur capacité de résoudre une variété de problèmes réels compliqués [98]. Dans le chapitre suivant, on discute l'algorithme d'optimisation basé sur l'enseignement-apprentissage classé dans la nouvelle génération des algorithmes à base de population.

Chapitre 2: L'algorithme d'optimisation basé sur l'enseignement-apprentissage

2.1. Introduction

Un nombre important de méthodes et d'algorithmes d'optimisation sont développés. Parmi eux, les algorithmes basés sur la population sont les plus populaires [8], car ils traitent un ensemble de solutions à chaque itération, ce qui permet de générer un ensemble de solutions de Pareto bien réparties aussi près que possible du véritable front de Pareto, plus rapidement que les méthodes classiques. Ce type utilise la stratégie d'amélioration itérative pour générer un ensemble de solutions proche du front de Pareto réel. Il comprend de nombreux algorithmes tels que les algorithmes génétiques (GAs), l'optimisation par colonies de fourmis (ACO), l'optimisation par essaims de particules (PSO), les colonies d'abeilles artificielles (ABC), l'optimisation de la recherche de bactéries (BFO), les algorithmes d'optimisation immunitaire (IAs) et l'optimisation basée sur l'enseignement-apprentissage (TLBO). L'algorithme TLBO récemment proposé s'inspire du processus d'enseignement-apprentissage en classe. Le premier TLBO mono-objectif est proposé par Rao et al [2], puis plusieurs variantes sont adaptées pour résoudre différents problèmes d'optimisation. Tels que : les problèmes continus [99], les problèmes binaires [100, 101], les problèmes discrets [102, 103], les problèmes à grande échelle [3], les problèmes multi-objectif [104, 105].

Le reste de ce chapitre est organisé comme suit. La section 2.2 présente le principe de base d'un algorithme d'optimisation basé sur le processus d'enseignement-apprentissage. La section 2.3 passe en revue les modifications et améliorations majeures proposées dans différentes variantes. Cette section discute aussi les hybridations de cet algorithme avec d'autres méthodes mathématiques et algorithmes à base de population. Alors que la section 2.4 se focalise sur les variantes et les modifications majeures de TLBO développées pour résoudre les problèmes multi-objectif. La section 2.5 représente les domaines d'application de l'algorithme TLBO. Alors que la section 2.6 discute les possibilités de recherches futures dans ce contexte.

2.2. Le principe de base d'un algorithme basé sur l'enseignement-apprentissage

Un algorithme d'optimisation basé sur l'enseignement-apprentissage est un algorithme à base sociale. C'est l'imitation du processus d'enseignement-apprentissage dans la classe, introduit sous sa forme mono-objectif par Rao et al. [2, 3]. Mais contrairement aux algorithmes traditionnels basés sur la population, l'algorithme Teaching-Learning Based Optimization (TLBO) se compose essentiellement de deux phases : la phase d'enseignement (Teaching phase) et la phase d'apprentissage (Learning phase), comme le montre la figure 2.1 et l'algorithme 2.1. Les apprenants obtiennent des connaissances non seulement à partir de l'enseignement dispensé par un enseignant, mais aussi à partir de leur interaction mutuelle [106] par le biais de discussions de groupe, de présentations et de communications formelles. Chaque individu de la population est considéré comme un apprenant et les variables sont considérées comme des sujets à apprendre. Après avoir évalué les apprenants, le meilleur apprenant est sélectionné pour devenir un enseignant pour la prochaine phase d'enseignement. Tandis que la phase d'apprentissage sert à échanger les connaissances entre les apprenants. Plusieurs variantes de TLBO ont besoin de paramètres communs seulement, telles que le critère d'arrêt (nombre d'évaluations ou nombre de générations) et la taille de la population.

Dans la phase d'enseignement, les apprenants apprennent par l'intermédiaire de l'enseignant. Ce dernier essaie d'améliorer les résultats de ses apprenants. Chaque apprenant X_i apprends auprès de l'individu le plus compétent de la population actuelle (enseignant), désigné par X^T, prenant en considération le niveau moyen de tous les apprenants X^{mean}. Au cours de cette phase, un nouvel apprenant X_i^{new} est obtenu en utilisant l'équation (2.1).

$$X_i^{new} = X_i + r_i(X^T - T_f X^{mean}) \tag{2.1}$$

$$X^{mean} = \frac{1}{N}\sum_{i=1}^{N} X_i \tag{2.2}$$

Où $r_i \epsilon$ [0,1] est un nombre aléatoire de la distribution uniforme ; T_f est le facteur d'enseignement $\epsilon \{1, 2\}$ pour souligner l'importance de la qualité moyenne des apprenants X^{mean}; et N est la taille de la population.

Si $F(X_i^{new})$ est meilleur que $F(X_i)$, alors X_i^{new} remplacera X_i, sinon il sera écarté.

Après l'achèvement de la phase d'enseignement, les apprenants actualisés participeront à la phase d'apprentissage. Un apprenant interagit avec un autre apprenant, pour augmenter ses connaissances si ce dernier a plus de savoir que lui. Pour chaque apprenant X_i, un collègue X_j est choisi au hasard pour interagir avec lui suivant les équations (2.3) et (2.4) où, $r_i \epsilon$ [0,1] est un nombre aléatoire.

$$\text{Si } X_j \text{ est meilleur que } X_i : X_i^{new} = X_i + r_i \times (X_j - X_i) \tag{2.3}$$

$$\text{Sinon } X_i^{new} = X_i + r_i \times (X_i - X_j) \tag{2.4}$$

La nouvelle solution X_i^{new} peut remplacer la solution actuelle X_i si $F(X_i^{new})$ est meilleure que $F(X_i)$, sinon elle sera écartée.

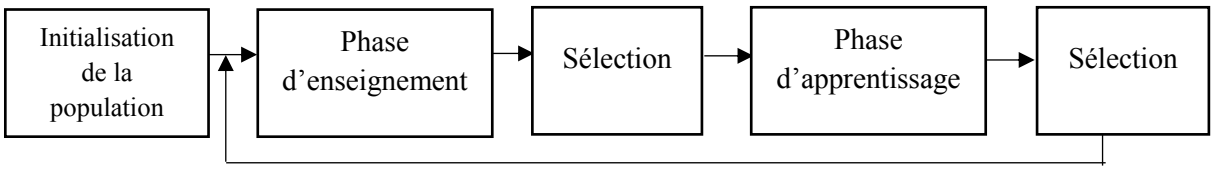


Figure 2.1 : Les étapes principales d'un algorithme TLBO

Algorithme 2.1 : Pseudo-code de l'algorithme TLBO de base

Les entrées : N le nombre d'individus (la taille de la population) ;
Les sorties : l'enseignant $\boldsymbol{X^T}$;

```
Initialisation aléatoire et évaluation des apprenants (individus)
Sélectionner X^T le meilleur apprenant et calculer la moyenne X^mean
Tant que la condition d'arrêt n'est pas atteinte
  Pour chaque apprenant                                   // phase d'enseignement
    Calculer T_f = round(1 + rand (0,1))
    Mettre à jour l'apprenant en utilisant l'équation (2.1)
  Fin pour
  Évaluer les nouveaux apprenants
  Accepter les nouveaux apprenants s'ils sont meilleurs que les anciens
  Pour chaque apprenant                                   // phase d'apprentissage
    Sélectionner un autre apprenant aléatoire différent
    Mettre à jour l'apprenant en utilisant l'équation (2.3) ou (2.4)
  Fin pour
  Évaluer les nouveaux apprenants
  Accepter les nouveaux apprenants s'ils sont meilleurs que les anciens
  Mettre à jour l'enseignant et la moyenne X^mean
Fin tant que
```

2.3. Les améliorations et hybridations de TLBO

Depuis son introduction par Rao et al. [2], en 2011, plusieurs modifications sont faites sur cet algorithme afin d'améliorer ses performances [8]. Les différentes variantes de TLBO peuvent être classées selon : les améliorations de processus d'enseignement-apprentissage et les hybridations avec d'autres méthodes d'optimisation.

2.3.1. Les algorithmes TLBO améliorés

Dans ce qui suit, quelques modifications majeures liées au TLBO sont passées en revue :

- **L'apprentissage oppositionnel et quasi-oppositionnel** : Dans les travaux [107, 108, 109] et d'autres, la population initiale est générée en utilisant une technique appelée l'apprentissage oppositionnel/quasi-oppositionnel [8]. Une population de N apprenants est générée d'une manière aléatoire ensuite une population opposée (quasi-opposée) de ces apprenants est créée suivant le principe décrit sur la figure 2.2. Ensuite, les N meilleurs apprenants entre ces deux populations sont sélectionnés comme population initiale.

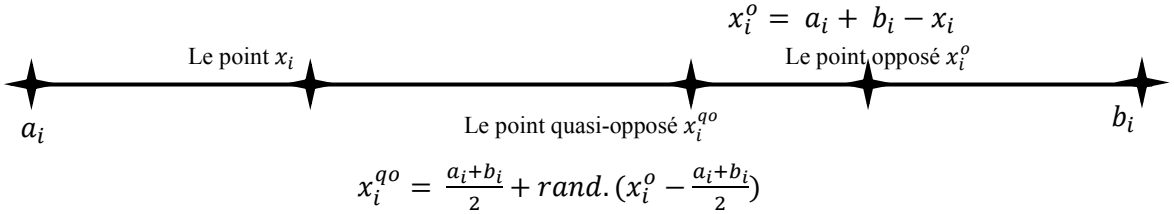


Figure 2.2 : L'apprentissage oppositionnel et quasi-oppositionnel

- **Paramètres adaptatifs** : Généralement, c'est une bonne idée d'encourager les apprenants à découvrir les sujets divers au cours des premières étapes de la recherche. Au cours des étapes ultérieures, il faut ajuster finement les connaissances des apprenants afin qu'ils soient plus experts. Suivant cette stratégie et selon le brief commun que l'exploration est faite au cours des premières étapes et l'exploitation est plus activée dans les étapes ultérieures, Cao et al. [110] ont introduit un facteur de pondération adaptatif ajustant les mouvements des individus. La notion

de poids a été introduite aussi dans [111] et [112] pour simuler le phénomène d'oubli d'une partie de connaissances requises ; seuls quelques individus peuvent atteindre leur objectif. Tandis que Li et al. [113] ont proposé d'utiliser un poids d'inertie ajustant l'influence de l'apprenant et un coefficient d'accélération ajustant la taille du pas d'apprentissage. Alors que dans [114], les valeurs aléatoires dans les équations (2.1) (2.3) et (2.4) sont remplacées par la constante d'enseignement/apprentissage ajustée, dynamiquement et respectivement, en fonction de la fitness de l'apprenant, la génération actuelle et la génération maximale.

Un système à population variable a été proposé dans [115], où la taille de la population varie durant le processus de recherche. Pour la diminuer, les individus les plus similaires seront éliminés alors que pour augmenter la taille de la population de nouveaux individus seront générés avec une variance et une moyenne déterminées [8]. Tandis que dans [116], la taille de la population participante dans chaque phase est proportionnelle à la moyenne des solutions réussites dans la phase en question.

- **Sélection et remplacement** : Dans [116], une nouvelle solution générée dans les phases d'enseignement et d'apprentissage sera un membre de la nouvelle population lorsque sa fitness est supérieure à la moyenne des apprenants de la population précédente. Rao et Patel [117] ont proposé de choisir les meilleures solutions comme solutions d'élite à chaque génération. Après avoir passé le processus d'apprentissage (enseignement et apprentissage), les Y pires individus de la population sont remplacés par les individus d'élite et une stratégie de mutation des doublons est appliquée (voir la figure 2.3).

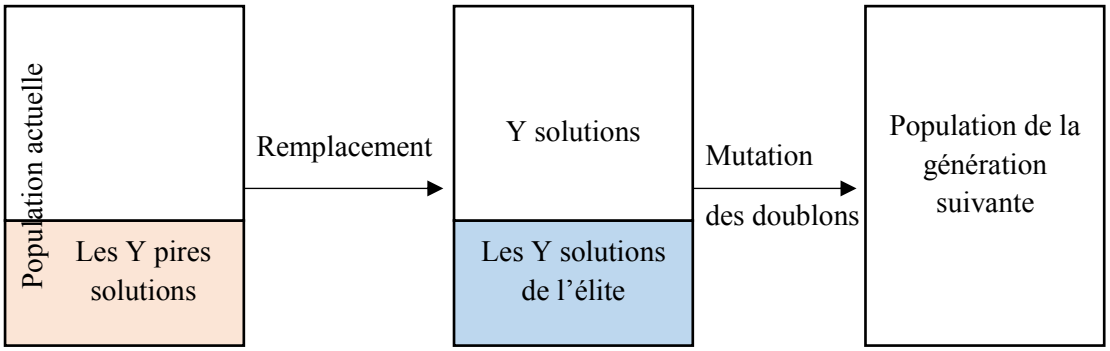


Figure 2.3 : Remplacement des pires apprenants par les meilleurs trouvés durant le processus de recherche

- **Phases supplémentaires** : Dans [118], une phase d'autoformation est exécutée par les enseignants pour augmenter la capacité de recherche locale. Alors que dans [119], la phase d'apprentissage est remplacée par deux phases : la phase d'apprentissage mutuel et la phase d'auto-apprentissage. Dans la variante fournie dans [120], deux phases supplémentaires sont introduites dans l'algorithme TLBO basique : une phase d'apprentissage par auto-rétroaction (auto-feedback) après la phase d'enseignement, et une phase de mutation et croisement après la phase d'apprentissage pour améliorer l'exploration de l'algorithme. Après les deux phases conventionnelles, une phase appelée chaotique est introduite par Farah et al. [121], un nouvel individu perturbé est généré de chaque apprenant existant.
- **Stratégies intégrées** : Différentes stratégies sont utilisées dans les deux phases. Krishna et Sao [122] ont introduit l'apprentissage tutoriel et l'apprentissage autonome, respectivement, dans la phase d'enseignement et d'apprentissage. Une phase d'apprentissage auto-motivé et une phase d'apprentissage tutoriel sont considérées dans la phase d'apprentissage [123]. Raja et al. [124] ont proposé une variante de TLBO où la méthode de formation par tutoriel et la méthode d'auto-apprentissage sont incorporées respectivement dans la phase d'enseignement et la phase d'apprentissage. Alors que dans LETLBO (TLBO with Learning Experience of others) [125],

les apprenants dans les deux phases utilisent des possibilités aléatoires pour sélectionner la stratégie originale ou bien la méthode d'apprentissage basée sur les expériences des autres (voir la figure 2.4).

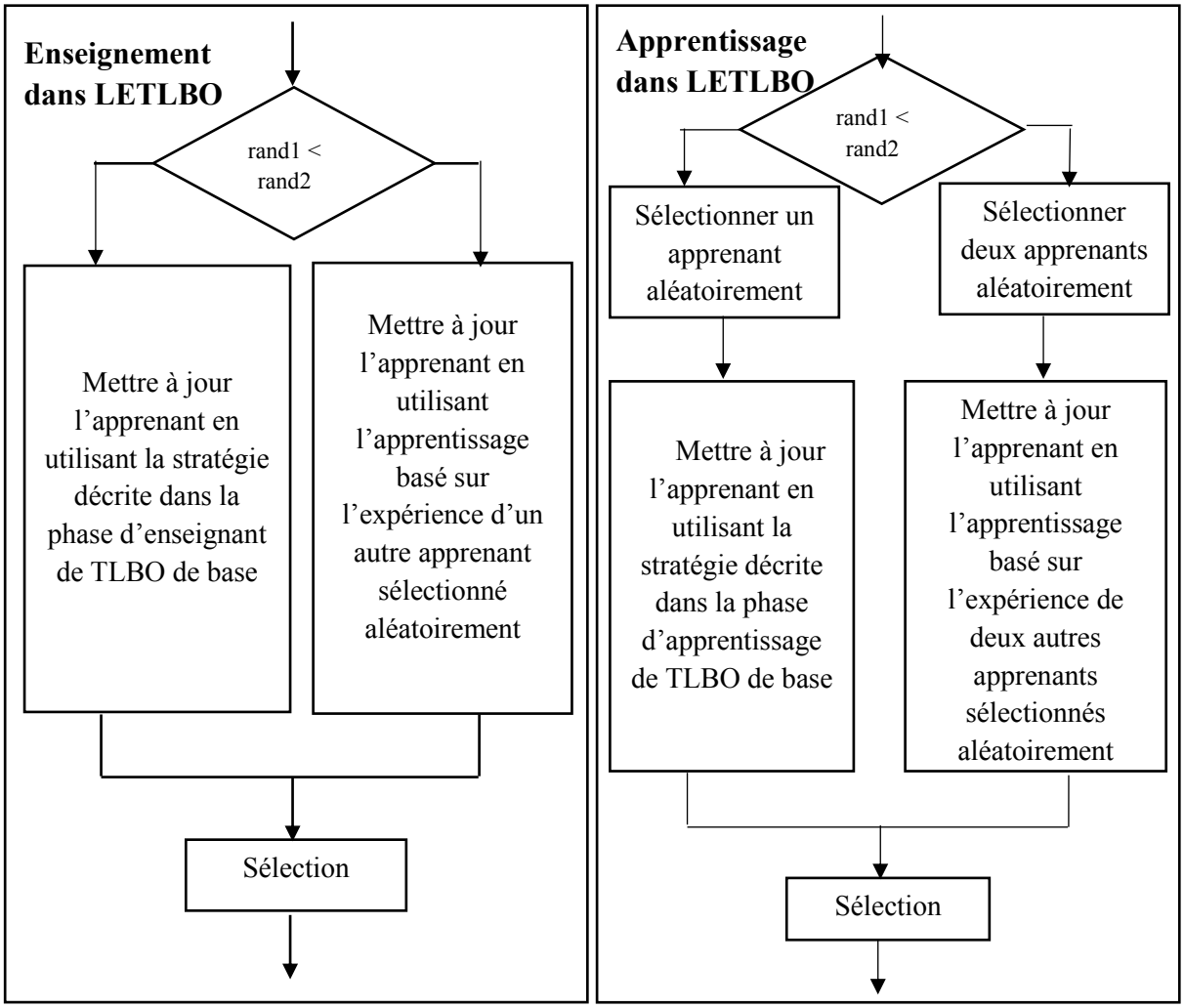


Figure 2.4 : L'organigramme des deux phases modifies de la variante LETLBO

- **Partition de la population** : Dans [126] la population est divisée en k sous-populations selon la distance Euclidienne entre les apprenants. Alors que dans [127] et afin d'augmenter la diversité, les apprenants sont divisés en petits groupes dynamiques selon la distance euclidienne. Lors de la phase d'enseignement, les individus du même groupe sont affectés par la moyenne de leur groupe seulement. Une approche de regroupement adaptatif d'apprenants a été utilisée dans [128]. Où, les groupes sont formés en fonction de leur distribution dans l'espace de recherche. Une variante à deux niveaux hiérarchiques a été proposée dans [129]. Des sous-essaims sont construits d'une manière aléatoire et évolués de manière indépendante. Et les meilleurs apprenants de cette couche construisent un autre essaim dans la couche supérieure pour suivre un certain processus d'apprentissage (voir la figure 2.5). Après quelques itérations, les sous-essaims sont reconstruits.

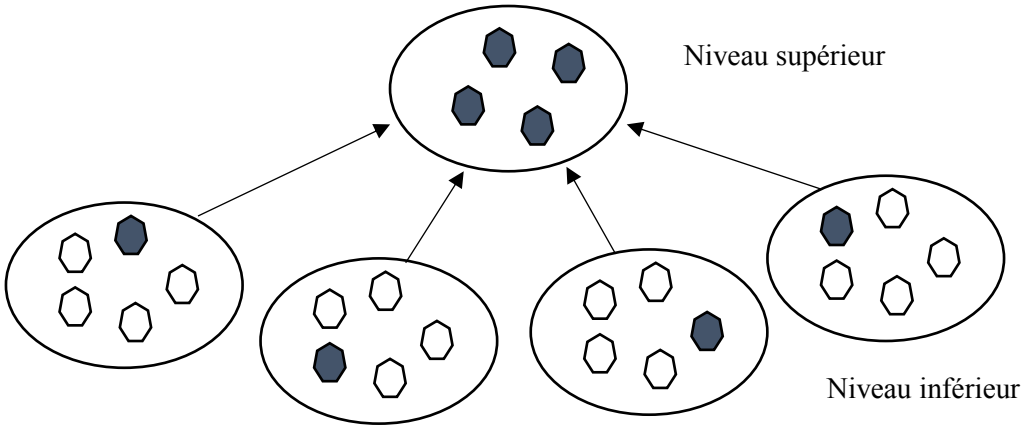


Figure 2.5 : Population multi-essaim avec deux niveaux hiérarchiques

2.3.2. Les algorithmes TLBO hybrides

L'algorithme d'optimisation basé sur l'enseignement-apprentissage (TLBO) a été combiné avec d'autres algorithmes d'optimisation et techniques de recherche. Le tableau 2.1 résume quelques exemples d'hybridations développées.

Tableau 2.1 : Exemples d'hybridations de TLBO avec d'autres algorithmes et techniques de recherche

	Combiné avec	**Référence**
Techniques de recherche	La fonction de recherche locale	[130]
	L'opérateur de recuit simulé	[131]
	La méthode de quasi-Newton	[132]
	La stratégie d'approximation stochastique à perturbation simultanée	[133]
	La méthode d'information sur l'expérience et mutation différentielle	[134]
	La topologie de voisinage en anneau	[135]
	L'apprentissage par l'échantillonnage gaussien	[136]
	La recherche tabou robuste	[137]
	La stratégie de correction d'erreur	[138]
	Des principes de la théorie des graphes	[139]
Algorithmes d'optimisation	L'algorithme d'optimisation par essaim de particules (PSO)	[140, 141]
	Évolution différentielle (DE)	[142]
	Algorithme génétique (GA)	[143]
	Algorithme de recherche par Coucou (CSA)	[144]
	Colonie d'abeilles artificielle (ABC)	[145]
	L'algorithme de la recherche d'harmonie (HS)	[146]
	Les réseaux de neurones	[147]
	Les réseaux de neurones flous	[148]

Par la suite, quelques travaux dans ce contexte sont classés selon le but de l'hybridation.

- **Améliorer la génération de la population initiale** : Une méthode d'initialisation en deux étapes TSI a été utilisée dans [149] pour l'initialisation de la population, ensuite le TLBO de base a été exécuté.

- **Accélérer la procédure de recherche** : l'algorithme d'optimisation hybride basé sur l'enseignement-apprentissage par essaim de particules multi-objectif (HTL-MOPSO) est proposé par Cheng et al. [140], afin d'améliorer et d'accélérer la capacité de la procédure de recherche. Car dans le MOPSO canonique, il y a un manque d'interaction et de partage d'informations entre les particules. Les phases d'enseignement et d'apprentissage apportent cette notion manquante au HTL-MOPSO.
- **Équilibrer la recherche locale et globale** : Dans [132], une méthode quasi-Newton a été intégrée à l'algorithme TLBO pour améliorer ses capacités de recherche locale. Où Qu et al. ont utilisé TLBO pour effectuer une recherche globale et la méthode quasi-Newton pour raffiner, en faisant la recherche locale. Dans [133], pour la même raison d'équilibrer la capacité d'exploration globale et l'exploitation locale, TLBO a été combiné avec une stratégie d'approximation stochastique à perturbation simultanée. Alors que Wang et al. [134] combinent les méthodes d'information sur l'expérience et la mutation différentielle avec TLBO. L'intégration de la première a été réalisée avant le processus d'apprentissage pour définir la direction de la recherche et donner des informations sur les tendances. Alors que la deuxième a été adoptée pour améliorer la diversité de la population. Dans l'algorithme hybride HTLBO-HS qui a été proposé dans [146], la recherche d'harmonie a été utilisée pour optimiser l'enseignant actuel. Alors qu'une méthode de recherche locale a été utilisée pour équilibrer les capacités d'exploration et d'exploitation. Généralement, les optimums globaux sont proches des optimums locaux. Pour cette raison, une topologie de voisinage en anneau a été introduite dans [135]. Chaque apprenant échange des informations avec ses deux voisins.
- **Améliorer la précision** : Le nombre aléatoire dans les deux phases a été remplacé par une distribution de Cauchy, dans [138]. Aussi une stratégie de correction d'erreur est utilisée. L'hybridation de TLBO avec ces deux stratégies a résolu le problème de précision et a amélioré la capacité d'auto-ajustement et antiblocage.

2.4. TLBO pour les problèmes multi-objectif

2.4.1. Le premier MOTLBO

Le premier algorithme TLBO multi-objectif est développé par Niknam et al [71]. Par la suite, plusieurs algorithmes multi-objectif sont proposés. Dans l'algorithme proposé, la mutation a été utilisée pour éviter la convergence prématurée. Alors que, le problème a été résolu avec la méthode mathématique Min-Max, aucun mécanisme de tri de solutions n'a été utilisé.

2.4.2. Adoption de tri non-dominé (NDS) et distance d'encombrement (CD)

Zou et al. [150] ont développé l'approche TLBO multi-objectif (MOTLBO) basée sur les deux mécanismes de tri non-dominé (NDS : Non-Dominated Sorting) et de distance d'encombrement (CD : crowding Distance), adoptés de l'algorithme génétique NSGA-II. Notons que le pseudo-code des deux mécanismes est présenté dans l'annexe 4. Comme le montre l'algorithme 2.2, une archive externe est utilisée pour stocker les meilleurs apprenants. Un enseignant est sélectionné parmi les archives externes, avec la plus grande distance d'encombrement. La nouvelle solution remplace l'ancienne dans deux cas, si elle la domine ou bien si elles sont incomparables. Dans le deuxième cas, une probabilité de replacement a été utilisée, en retournant une pièce de monnaie. Différemment de l'algorithme de base, le facteur d'enseignement varie entre 1 et 2 et le nombre aléatoire est exclu dans la phase d'apprentissage.

Algorithme 2.2 : Pseudo-code de l'algorithme MOTLBO [150]

Les entrées : N le nombre d'individus (la taille de la population), condition d'arrêt ;
Les sorties : l'archive A ;

Initialisation aléatoire et évaluation des apprenants
Trier les apprenants en appliquant le tri non-dominé et la distance d'encombrement
Conserver la population dans l'archive A
Tant que la condition d'arrêt n'est pas atteinte
  Calculer la moyenne X^{mean}
  Pour chaque apprenant X_i // phase d'enseignement
    Sélectionner un enseignant X^T du premier front avec la distance d'encombrement maximale
    Calculer $T_f = 1 + rand\ (0,1)$
    Mettre à jour l'apprenant X_i en utilisant l'équation (2.1)
    Évaluer le nouvel apprenant X_i^{new}
    Si X_i^{new} domine X_i **alors** remplacer X_i par X_i^{new}
    Sinon Si X_i et X_i^{new} sont incomparables et rand < 0.5 **alors** remplacer X_i par X_i^{new}
      Fin si
    Fin sinon
  Fin pour
  Pour chaque apprenant X_i // phase d'apprentissage
    Sélectionner un autre apprenant X_j différent de X_i
    Si X_j domine X_i **alors**
      Mettre à jour l'apprenant en utilisant $X_i^{new} = X_i + (X_j - X_i)$
    Sinon Mettre à jour l'apprenant en utilisant $X_i^{new} = X_i + (X_i - X_j)$
    Fin sinon
    Évaluer le nouvel apprenant X_i^{new}
    Si X_i^{new} domine X_i **alors** remplacer X_i par X_i^{new}
    Sinon Si X_i et X_i^{new} sont incomparables et rand < 0.5 **alors** remplacer X_i par X_i^{new}
      Fin si
    Fin sinon
  Fin pour
  Mélanger la population avec l'archive A
  Trier le tout en appliquant le tri non-dominé et la distance d'encombrement
  Sélectionner les N meilleurs apprenants parmi 2N triés
  Mettre à jour la population et l'archive
Fin tant que

2.4.3. Apprentissage auprès de plusieurs enseignants

Dans le même contexte d'adoption de mécanismes pour trier les solutions générées, Lin et al. [151] ont utilisé la méthode de construction d'un ensemble non-dominé proposé dans [152] au lieu de tri non-dominé de NSGA-II. Seules les solutions du premier front figurent dans cet ensemble. Chaque apprenant X_i est comparé à chaque apprenant X_j dans l'ensemble non-dominé. Si X_j est dominé par un apprenant X_i, X_j sera supprimé de l'ensemble. Si l'apprenant X_i n'est pas dominé par aucun apprenant de l'ensemble non-dominé, ce dernier sera ajouté à cet ensemble. Afin de sélectionner les enseignants, la distance d'encombrement a été calculée entre les solutions de l'ensemble non-dominé, seulement, ce qui réduit la complexité de l'algorithme.

Dans l'algorithme MOTLBO proposé dans [151] (voir la figure 2.6), l'apprentissage dans la phase d'enseignement est effectué par plusieurs enseignants $X_j^T, j \in \{1,2, \dots k\}$, l'apprenant X_i est mis à jour suivant l'équation (2.5). Sachant que $T_{f,j}$ est calculé par les équations (2.6) et (2.7)

$$X_i^{new} = X_i + \sum_{j=1}^{k} r_{i,j}.(X_j^T - T_{f,j}\, X^{mean}) \quad (2.5)$$

Alors que, dans la phase d'apprentissage, pour chaque apprenant X_i, un autre apprenant X_j différent est sélectionné aléatoirement. L'apprenant est mis à jour suivant le même principe de l'algorithme proposé par Zou et al [150], en utilisant les équations de l'algorithme de base (2.3) et (2.4). Tandis que l'opérateur de sélection est comme suit : Les deux solutions $X_i \; et \; X_i^{new}$ sont comparées au sens de Pareto. Si X_i^{new} domine X_i, elle sera ajoutée à la population et X_i sera déposée. Si les deux sont égales au sens de Pareto et suivant le même principe dans [150], une pièce de monnaie sera retournée pour déterminer la solution acceptée.

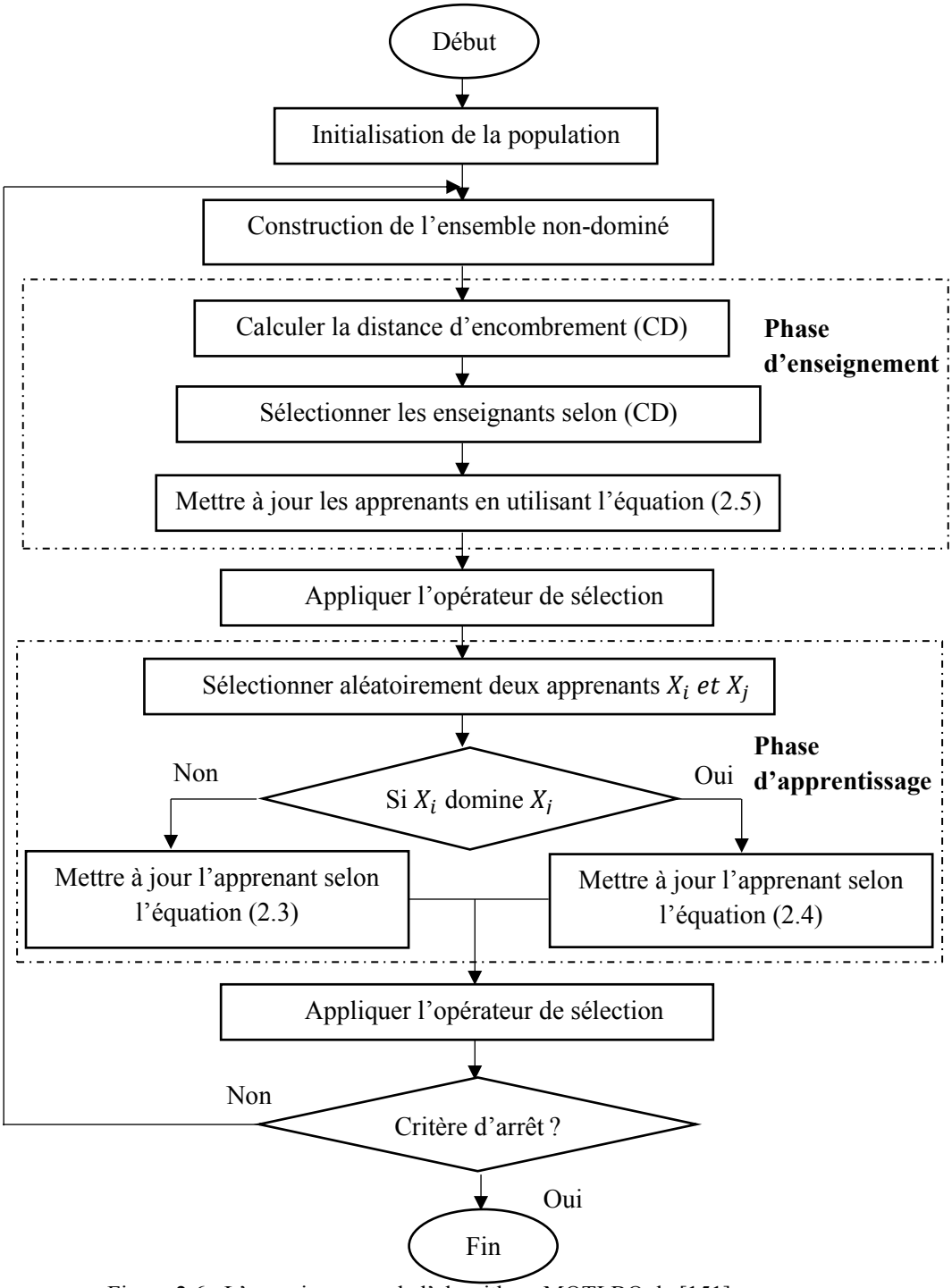


Figure 2.6 : L'organigramme de l'algorithme MOTLBO de [151]

2.4.4. Facteur d'enseignement adaptatif

Dans l'algorithme TLBO de base, le facteur d'enseignement peut être 1 ou 2. Suivant le principe de « tout ou rien », dans le premier cas l'apprenant n'apprend rien auprès de son enseignant et dans le deuxième il apprend toutes les connaissances de son enseignant. Cette pratique ne représente pas le phénomène réel du processus, où les apprenants peuvent apprendre des connaissances avec des proportions différentes et qui varient du « tout » vers « rien ».

De ce fait, Rao & Patel [153, 154] ont proposé un facteur d'enseignement adaptatif. Les grandes valeurs de ce facteur accélèrent la recherche mais ne permettent pas une bonne exploration de l'espace ce qui peut provoquer une convergence prématurée vers des optimums locaux. Alors que, les petites valeurs permettent une recherche plus fine à petits pas mais provoquent une convergence plus lente. Pour chaque enseignant X_j^T, un facteur de $T_{f,j}$ est calculé en se basant sur la moyenne X^{mean}, suivant l'équation (2.6). Alors que dans [151], le facteur est calculé par l'équation (2.6) et normalisé par la suite suivant l'équation (2.7).

$$T_{f,j} = \frac{X^{mean}}{X_j^T}, j = 1,2,\dots,k \tag{2.6}$$

$$T_{f,j} = \begin{cases} 2, si\ T_{f,j} > 2 \\ T_{f,j}, si\ 1 \le T_{f,j} \le 2 \\ 1, si\ T_{f,j} < 1 \end{cases} \tag{2.7}$$

Un facteur d'enseignement adaptatif T_f qui dépend des résultats de l'apprenant i et l'enseignant de son groupe j est utilisé dans [105]. Dans ce travail, et à la différence des deux travaux précédents, le facteur est calculé dans l'espace objectif et non pas l'espace de recherche par l'équation suivante :

$$T_{f,i\,j} = \begin{cases} \frac{f(X_i)}{f(X_j^T)} & si\ f(X_j^T) \neq 0 \\ 1 & sinon \end{cases} \tag{2.8}$$

2.4.5. Sélection auto-adaptative de l'approche d'apprentissage

Un algorithme TLBO multi-objectif auto-adaptatif (SA-MTLBO) est proposé dans [155, 156]. Pour chaque apprenant X_i, l'approche d'apprentissage (une des deux phases) est sélectionnée de manière auto-adaptative, selon son niveau de connaissances défini par son rang. L'apprenant choisit une des deux phases à suivre à chaque itération. Une probabilité P_i est calculée en fonction du rang de l'apprenant et la taille de la population. Le tri non-dominé et le calcul de distance d'encombrement sont adoptés pour obtenir une relation d'ordre total entre les solutions suivant la règle : en comparant deux solutions, une solution est rangée i et l'autre est rangée j tel que $i < j$, si $front\ (i) < front\ (j)$ ou bien $front\ (i) = front\ (j)\ et\ CD(i) > CD(j)$. Sachant que N est la taille de la population, chaque solution i obtient un rang R_i défini par :

$$R_i = N - i, i = 1,2,\dots N \tag{2.9}$$

Selon l'équation (2.9), les meilleures solutions obtiennent un rang R_i plus élevé. Par la suite, la probabilité de sélection $P_i \in [0,1]$ est calculée par :

$$P_i = 0.5\left(1 - \cos\left(\frac{R_i \pi}{N}\right)\right), i = 1,2,\dots,N \tag{2.10}$$

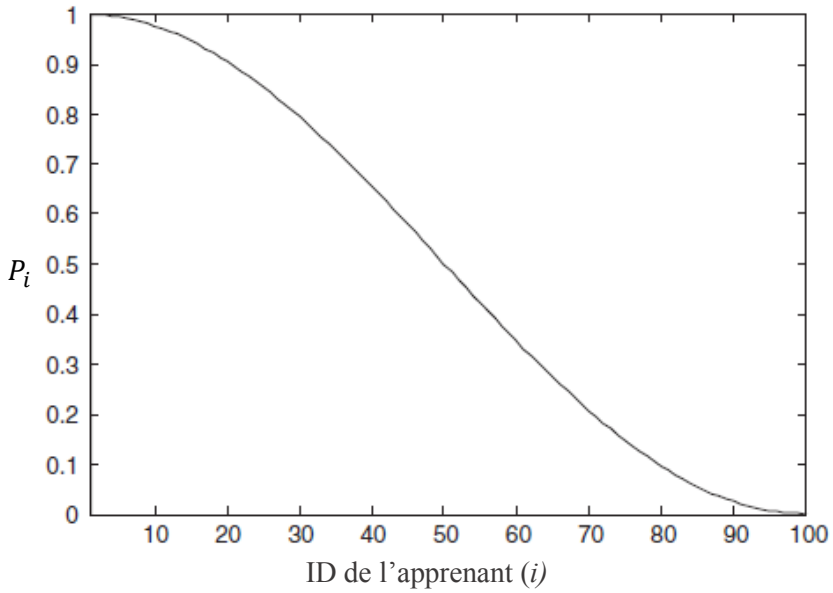


Figure 2.7 : La probabilité de sélection P_i de 100 apprenants rangés [155].

Après avoir calculé la probabilité, la méthode de sélection de l'approche d'apprentissage (la phase) est appliquée suivant l'algorithme (2.3). Les meilleurs apprenants sont plus susceptibles de choisir la phase d'apprentissage, ce qui améliore la diversité de la population. Et pour améliorer la capacité de convergence, les autres apprenants sont plus susceptibles de choisir la phase d'enseignement. Les résultats prouvent que cette méthode de sélection améliore les performances de l'algorithme, comparé à ceux de l'exécution en série des deux phases.

Algorithme 2.3 : Sélection auto-adaptative de la phase

Les entrées : la population ;
Les sorties : appel à une phase ;

```
Assigner les apprenants à des fronts en utilisant NS
Calculer la distance d'encombrement (CD)
Assigner un rang R_i à chaque solution i en utilisant l'équation (2.9)
Calculer la probabilité P_i de chaque solution i en utilisant l'équation (2.10)
Pour i = 1 à N
  Si rand ≥ P_i alors
    Employer la phase d'enseignement
  Sinon
    Employer la phase d'apprentissage
  Fin si
Fin pour
```

2.4.6. Apprentissage en groupe

Afin d'augmenter les capacités d'exploration et exploitation de l'algorithme, les auteurs de [153, 154, 105] ont utilisé une technique d'apprentissage en groupe. Les apprenants sont divisés en groupes, un enseignant est sélectionné pour chaque groupe. La moyenne du groupe est utilisée dans la phase d'enseignement, au lieu de la moyenne de la population. Si les résultats d'un groupe sont satisfaisants, ce groupe est attribué à un meilleur enseignant. Le problème bi-objectif dans les travaux présentés est transformé en un problème mono-objectif $f(X)$. L'enseignant en chef X_c est celui ayant la meilleure fitness $T_1 = f(X_c)$. Les autres $(k-1)$ enseignants sont sélectionnés en se basant sur la fitness de l'enseignant en chef suivant l'équation (2.11). Par la suite, les apprenants sont affectés à leurs enseignants, suivant l'algorithme (2.4).

$$T_s = f(X_c) \pm r_i * f(X_c)\ s = 2,3,\dots,k \tag{2.11}$$

Algorithme 2.4 : Affectation des apprenants à des enseignants

Les entrées : les apprenants, les enseignants ;
Les sorties : groupes d'apprenants ;

```
Pour i = 1 à N          % taille de la population
   Si T1 ≤ f(Xi) < T2
      Affecter l'apprenant Xi à l'enseignant 1
   Sinon
      Si T2 ≤ f(Xi) < T3
         Affecter l'apprenant Xi à l'enseignant 2
      Sinon
         ……..
         Affecter l'apprenant à l'enseignant k
      Fin si
   Fin si
Fin pour
```

2.4.7. L'apprentissage par tutoriel

Pendant les heures de tutorat, les apprenants peuvent également apprendre en discutant avec leurs camarades de classe et leurs enseignants. De ce fait, un mécanisme d'apprentissage par tutoriel est incorporé dans la phase d'enseignement. Les apprenants sont mis à jour en appliquant l'équation (2.12). Tel que l'apprenant X_j est sélectionné aléatoirement.

$$X_i^{new} = \begin{cases} X_i + r_{i1}(X^T - T_f X^{mean}) + r_{i2}(X_i - X_j) & si\ f(X_i) < f(X_j) \\ X_i + r_{i1}(X^T - T_f X^{mean}) + r_{i2}(X_j - X_i) & sinon \end{cases} \tag{2.12}$$

2.4.8. L'auto-apprentissage

Suivant l'idée de la possibilité d'existence d'une motivation chez les apprenants pour s'auto-apprendre afin d'améliorer leurs connaissances, une phase d'auto-apprentissage est ajoutée au processus après la phase d'apprentissage en utilisant l'équation (2.13) [153]. Où $X_{max}\ et\ X_{min}$ sont les limites supérieures et inférieures de l'espace de recherche. Alors que dans [105], l'auto-apprentissage est ajouté à la phase d'apprentissage avec un facteur d'exploration aléatoire $F_e = round(1 + rand)$, pour améliorer les connaissances avec l'aspect supplémentaire d'auto-apprentissage. Les apprenants sont mis à jour dans la phase d'apprentissage en appliquant l'équation (2.14).

$$X_i^{new} = X_i + r_i(X_{max} - X_{min}) \tag{2.13}$$

$$X_i^{new} = \begin{cases} X_i + r_i\left(X_i - X_j\right) + r_i(X^T - F_e\ X_i) & si\ f(X_i) < f(X_j) \\ X_i + r_i\left(X_j - X_i\right) + r_i(X^T - F_e\ X_i) & sinon \end{cases} \tag{2.14}$$

2.4.9. Instruction individualisée

Inspiré par l'éducation moderne, où l'apprentissage est devenu plus flexible, le processus d'apprentissage doit répondre aux besoins des individus pour les aider à s'améliorer. Des modifications majeures sont faites dans l'algorithme multi-objectif à instruction individualisée TLBO (INM-TLBO) [98]. Les mécanismes de tri non-dominé et de distance d'encombrement sont utilisés dans cet algorithme. Mais ils ont proposé un nouveau mécanisme de sélection appelé instruction individualisée, où la population est divisée en leaders avec un rang égal à 1 et tous les autres sont considérés comme des suiveurs. La modification centrale est l'opérateur de désignation, où chaque suiveur est affecté à son chef approprié en fonction de ses besoins spécifiques. La distance entre un suiveur et les leaders, qui le

dominent et qui ont un rang minimal, est calculée dans l'espace de décision. Ensuite, le plus proche est choisi pour être celui qui convient à ce suiveur. Les suiveurs qui partagent le même leader sont considérés comme un groupe [98]. Une sélection selon le principe de la roue à roulette est utilisée pour sélectionner les objets interactifs appropriés pour la phase d'apprentissage ou bien choisir l'enseignant de leaders. Une solution avec une plus grande distance d'encombrement a plus de chances de participer comme un objet interactif / enseignant de leaders.

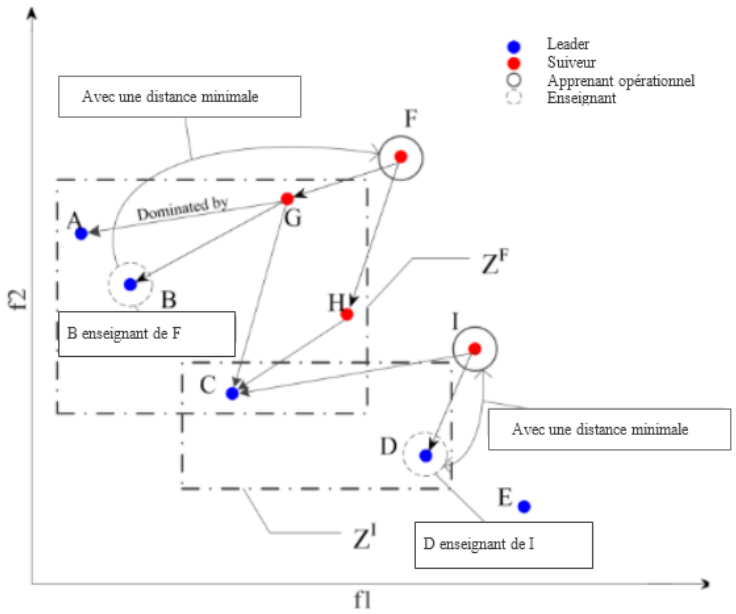


Figure 2.8 : L'opérateur de désignation d'enseignant dans INM-TLBO [98]

Comme le montre la figure 2.8, l'ensemble de solutions dominantes de l'apprenant F Z^F contient cinq apprenants (A, B, C, G et H). Les apprenants ayant le rang non-dominé le plus petit (solution du premier front) sont A, B et C. Parmi eux, il est supposé que l'apprenant le plus proche à F dans l'espace de décision est B. Ce dernier est désigné comme enseignant de l'apprenant F. De la même manière, D est désigné comme enseignant de l'apprenant I. Pour la désignation de l'objet interactif, représentée sur la figure 2.9, les apprenants A et E ont la plus grande probabilité pour être sélectionnés pour participer comme des objets interactifs, dans la phase d'apprentissage. À la différence de la roue à roulette pour la désignation de l'objet interactif, celle de la désignation de l'enseignant des leaders contient cinq choix seulement, qui sont les leaders (A, B, C, D, E).

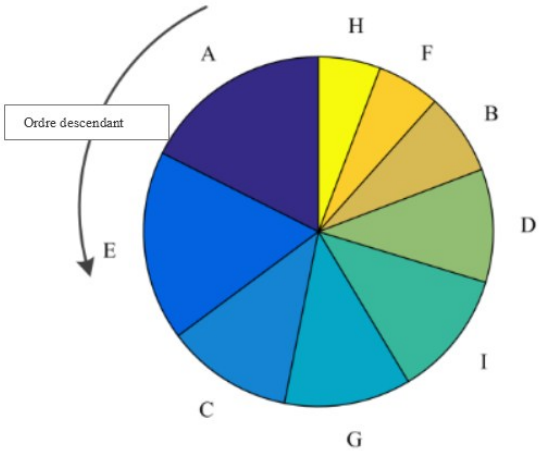


Figure 2.9 : L'opérateur de désignation de l'objet interactif de INM-TLBO [98]

2.5. Domaines d'application de l'algorithme TLBO

Quoique TLBO est relativement nouveau par rapport à d'autres algorithmes basés sur la population, et comme le montre le tableau 2.2, l'algorithme et ses variantes sont largement appliqués dans divers domaines.

Tableau 2.2 : Quelques domaines d'application de l'algorithme TLBO

Domaine	Description de l'application	Référence
Énergie	TLBO a été utilisé pour résoudre le problème du flux d'énergie.	[157, 158]
	ITLBO a été proposé pour maximiser les suiveurs de points d'énergie dans un système photovoltaïque.	[159]
	TLBO est appliqué afin de gérer l'énergie dans un micro-réseau.	[160]
Traitement d'image	L'algorithme a été modifié pour l'amélioration de l'image.	[161]
	TLBO a été proposé pour la compression des images satellitaires.	[162]
Médecine	Un algorithme TLBO basé sur les réseaux de neurones flous a été proposé pour estimer l'âge dentaire.	[148]
	Une version hybride a été proposée pour supprimer le bruit dans les images IRM	[163]
Robotique	Une variante hybride a été présentée pour résoudre le problème de navigation des robots mobiles.	[164]
	Deux variantes de TLBO ont été utilisées dans la conception optimale d'un robot manipulateur.	[165]
Les réseaux informatiques	Une variante de TLBO a été proposée pour résoudre un problème de localisation distribuée sans portée pour les réseaux de capteurs sans fil	[166]
	Des principes de la théorie des graphes ont été utilisés avec TLBO pour optimiser la configuration d'un cyber-réseau.	[139]
Data Mining	Une version de TLBO est utilisée pour la classification des données.	[167]
	TLBO a été combiné avec les réseaux de neurones pour résoudre un problème de classification de données.	[147]
Gestion et économie	Un modèle hybride a été proposé pour une allocation optimale du budget du gouvernement.	[168]
Conception et production	Un TLBO-observateur a été proposé pour optimiser la conception de structures.	[169]
	2S-TLBO basé sur deux stages a été utilisé pour résoudre le problème d'ordonnancement flexible dans les ateliers de production.	[170]

2.6. Les enjeux suspendus et les futures tendances de TLBO

TLBO dispose d'un fort potentiel pour résoudre différents problèmes. Malgré sa nouveauté il a réussi à ouvrir un petit monde dans le domaine d'optimisation [106]. Selon [106], les possibilités de recherches futures sur TLBO peuvent être résumées dans les points suivants :

- **Le réglage de paramètres** : La définition des paramètres affecte l'efficacité et la précision de l'algorithme. Les performances de l'algorithme peuvent être améliorées par l'ajustement approprié des paramètres. Comme nous l'avons déjà mentionné, cet algorithme est simple et facile à mettre en œuvre et avec peu de paramètres : la taille de la population, le facteur d'enseignement et le facteur aléatoire. Dans TLBO de base, le facteur d'enseignement T_f est égal à 1 ou 2. Ce qui cause une perte possible de la diversité et peut munir la recherche vers un échec ou bien une convergence prématurée.
- **Renforcement de bases théoriques** : Le comportement de l'algorithme TLBO a été toujours compris à partir de ses résultats expérimentaux et il n'a pas de base mathématique. Le développement d'un ensemble de théories mathématiques permet d'étudier l'impact de changer des paramètres et les ajuster dynamiquement selon la taille du problème par exemple. Ce qui facilite l'étude et la résolution d'un grand nombre de problèmes.
- **Réduction de la complexité** : TLBO est devenu l'un des fameux algorithmes d'optimisation modernes et de plus en plus de nouvelles variantes apparaissent et continuent à apparaitre. Comme nous l'avons déjà discuté, l'algorithme de base a connu plusieurs améliorations et hybridations avec d'autres algorithmes afin d'augmenter ses performances. Mais cette émergence a conduit vers une augmentation de la complexité et perte de simplicité de l'algorithme de base. D'où la nécessité d'assurer la simplicité et la convergence rapide dans les futures variantes à proposer.
- **Traitant les problèmes complexes et réels** : Avec le développement technologique et industriel, un grand nombre de problèmes complexes à grande échelle ont émergé. Alors qu'une seule variante [3] a été développée dans ce contexte. Passant en revue les travaux réalisés en utilisant TLBO, un grand nombre d'entre eux a été dédié à des problèmes continus. Alors que la plupart des problèmes dans la vie réelle sont discrets.

2.7. Conclusion

Dans ce chapitre, nous avons passé en revue les notions les plus importantes pour comprendre le fonctionnement de l'algorithme d'optimisation basé sur l'enseignement-apprentissage. Plusieurs améliorations sont apportées et des mécanismes très intéressants sont adoptés. TLBO a montré des capacités prometteuses pour résoudre différents problèmes, parmi eux, les problèmes multi-objectif qui nous intéressent. L'étude bibliographique faite dans ce chapitre nous a motivé pour développer une variante de TLBO permettant de résoudre le problème de regroupement d'objets bi-objectif. Cette approche sera discutée dans le chapitre 6.

Chapitre 3: La robotique en essaim

3.1. Introduction

Un système multi-robot a une variété d'avantages par rapport au robot monolithique [171]. Premièrement, la conception d'un robot monolithique capable d'accomplir différentes tâches, dans des conditions environnementales variées, est difficile [171]. En plus, un robot monolithique a une faible tolérance aux fautes [172, 173] (non robuste). Une petite panne d'un capteur peut empêcher l'accomplissement de la tâche. D'autre part, un système multi-robot s'adapte aux changements. Il peut bénéficier de l'hétérogénéité de ses individus pour accomplir des tâches qui nécessitent différentes capacités, et de son homogénéité pour couvrir la panne de certains individus. Une branche théorique et méthodologique, appelée robotique en essaim, pousse l'approche multi-robot à son extrémité, concevant un système multi-robot « intelligent » [172] avec des robots beaucoup moins intelligents, voire naïfs.

La robotique en essaim attire de plus en plus l'attention de la communauté de recherche [172], en s'inspirant des comportements intelligents observés chez les insectes sociaux tels que les fourmis, les abeilles, les termites [174] et d'autres systèmes **auto-organisés**[1] tels que les oiseaux, les poissons et les mammifères [172]. Un essaim a l'avantage de trouver son équilibre (de s'autoorganiser) même si quelques individus sont tombés en panne, d'autres individus sont intégrés ou bien les conditions de l'environnement sont changées. Ces propriétés sont appelées respectivement scalabilité, robustesse et flexibilité [173, 175, 176]. Les chercheurs ont montré que le système en essaim surmonte le système monolithique ou bien le système multi-robot général par rapport à ces 3 caractéristiques. Pour ces raisons et d'autres, ce domaine de recherche reste toujours très actif et de plus en plus adopté dans l'industrie [174].

Le reste de ce chapitre est organisé comme suit. La section 3.2 discute la source d'inspiration et la motivation pour laquelle la robotique en essaim a été étudiée. Les notions de base nécessaires pour comprendre ce domaine sont discutées dans la section 3.3. Dans la section 3.4, on décrit les trois propriétés caractérisant le système en essaim. Et dans la section 3.5, la taxonomie la plus concise est présentée. Ensuite, quelques scénarios étudiés dans la robotique en essaim sont présentés dans la section 3.6. Alors que dans la section 3.7 et la section 3.8, on s'est focalisé sur le problème traité dans ce travail en présentant les travaux proposés pour résoudre le problème d'énergie et le problème de regroupement d'objets, respectivement. Dans la section 3.9, les simulateurs et les robots physiques les plus récents et/ou les plus utilisés dans la robotique en essaim sont exposés. Tandis que la section 3.10 présente les applications potentielles et les limitations de la robotique en essaim.

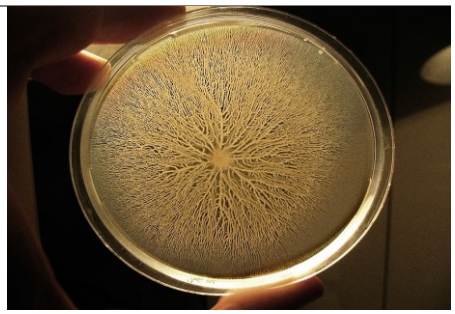
[1] L'auto-organisation est la combinaison de 4 règles principales : feedback positif, feedback négatif, l'aléatoire, et les interactions multiples [199], voir section 3.3.7

3.2. Motivation et sources d'inspiration

Plusieurs comportements observés dans la nature ont été considérés comme des aspects étranges et mystérieux [175], ce qui a poussé les chercheurs à étudier les systèmes naturels produisant ces comportements. Au cours des dernières décennies, ils ont démontré que la production de ces comportements extraordinaires [177] ne nécessite pas des connaissances individuelles sophistiquées, une planification ou bien un chef d'équipe. Les systèmes naturels étudiés, tels que les fourmis, les abeilles, les oiseaux, les poissons, etc. [172, 174], sont devenus par la suite la source d'inspiration d'une grande variété d'algorithmes tels que [178, 179, 180, 181]. Les individus ne sont pas informés du statut global de l'organisme, et leurs comportements ne sont pas guidés par un plan ou un chef [175]. Un individu est incapable d'accomplir la tâche tout seul, à cause de ses connaissances et capacités limitées.

Ces systèmes naturels peuvent accomplir des tâches vues au niveau macro comme étant des tâches complexes, comparées à la simplicité de l'individu, telles que le suivi de sentier des fourmis, la danse des abeilles mellifères, la construction de la termitière, la construction du nid de guêpes, bancs de poissons et flockage d'étourneaux[2] [175, 177]. Ils sont aussi capables d'échanger des informations sur la localisation de la nourriture et la présence d'un danger, tout en utilisant une interaction locale [173] implicite à travers les changements faits dans l'environnement, appelée **stigmergie**[3] [175]. Chaque individu modifie son comportement selon les changements précédents de ces collègues. Après une certaine quantité de changements et d'interactions locales, une organisation émergente auto-organisée est observée**.**

Tableau 3.1 : Quelques comportements en essaim observés dans la nature

Le système naturel	Le comportement en essaim	Exemple[4]
Les poissons	**Banc de poissons** [182] : Les poissons nagent ensemble, où chaque individu maintient, d'une manière continue, une zone de répulsion avec ses voisins pour éviter les collisions.	
Les bactéries	**Motilité d'essaimage** [183] : Des motifs dendritiques sont formés par la migration d'une population bactérienne sur une surface solide. Il est observé que l'essaim est plus résistant à certains antibiotiques [184].	

[2] Les étourneaux sont une famille très sociale des passereaux (classe d'oiseaux).
[3] La stigmergie est discutée dans la section 3.3.3
[4] Les photos utilisées ont les licences suivantes : les poissons (par Michio Morimoto, CC BY 2.0) ; les bactéries (par Adrian Daerr, CC BY-SA 4.0) ; les étourneaux (par Walter Baxter, CC BY-SA 2.0) ; les abeilles (par Nilfanion, CC BY-SA 3.0) ; les fourmis (par Igor Chuxlancev, CC BY 4.0) ; les lucioles (par xenmate, CC BY 2.0) ; les termites (par jbdodane, CC BY 2.0). (En utilisant Wikimedia Commons).

Les étourneaux	**Flockage** [185] : C'est un mouvement collectif synchronisé ou chaque étourneau coordonne ses déplacements avec ceux de ses X voisins les plus proches. Le nombre X est lié à la forme du troupeau plus que sa taille.	
Les abeilles	**La fission de colonies** [186, 187] : Une reine et des ouvrières quittent la ruche mère en formant un cône suspendu inversé près de la ruche, en attendant que les abeilles éclaireuses trouvent un site de nidification[5].	
Les fourmis	**La construction des ponts** [188, 189] : Les fourmis construisent des ponts avec leurs corps afin de combler les trous dans le sentier de fourragement.	
Les lucioles[6]	**Les éclats synchronisés** [177] : Chaque luciole corrige son cycle selon son voisinage jusqu'à l'arrivée à une synchronisation totale.	
Les termites	**La construction du nid** [187] : Des nids immenses sont construits en commençant par déposer des petites quantités de matériaux de construction imprégnés de phéromone ; ce qui encourage d'autres termites à déposer leurs matériaux.	

[5] Les abeilles éclaireuses (scouts) sont les plus expertes du groupe. Pour indiquer la distance et la direction de l'endroit qu'elle a trouvé, l'éclaireuse danse afin de convaincre d'autres scouts de vérifier son endroit. Après avoir atteint un seuil d'abeilles qui prennent le même choix, ces derniers empêchent les autres de danser pour d'autres endroits (par des signaux d'arrêt ou bien en interférant leur dance) [187]. De cette manière, une décision collective est prise et tout le groupe s'envole vers cet emplacement.

[6] Les lucioles (fireflies) sont des insectes lumineux.

3.3. Définitions et caractéristiques

3.3.1. L'essaim

Un essaim est défini comme étant un groupe de robots non-intelligents (des agents réactifs) formant, en tant que groupe, un système intelligent, d'une manière imprévisible [190]. Şahin [176] définit la robotique en essaim comme :

> *« L'étude de la manière dont un grand nombre d'agents incorporés physiquement, relativement simples, peuvent être conçus de telle sorte qu'un comportement collectif désiré* ***émerge*** [7] *à partir des interactions locales entre les agents et entre les agents et l'environnement. »*

Afin de distinguer entre les systèmes de robots en essaim (SRE) et les autres systèmes multi-robots (SMR), un ensemble de caractéristiques qualifient ce type de systèmes [175, 176, 191] : (1) les agents doivent être autonomes et capables d'interagir et d'agir sur leur environnement ; (2) ces agents doivent être homogènes, et même s'il existe une hétérogénéité, le nombre de groupes ne doit pas être grand ; (3) la communication et la détection sont limitées dans un voisinage local, ce qui donne au système la capabilité d'être scalable puisque la coordination est distribuée dans ce cas (4) le nombre d'agents doit être suffisamment grand pour permettre la convergence du système ; (5) un seul agent est incapable d'accomplir la tâche ciblée, l'ensemble d'agents doit collaborer ; En plus, (6) le système de contrôle dans un essaim est décentralisé.

Tableau 3.2 : Comparaison entre les SMR et SRE [192]

	SRE	**SMR**
Taille de la population	Grande	Petite
Contrôle	Décentralisé et autonome	Centralisé et à distance
Homogénéité	Homogène/quasi-homogène	Hétérogène
Environnement	Inconnu	Connu/inconnu
Mouvement	Oui	Oui
Scalabilité [8]	Élevée	Faible
Flexibilité [8]	Élevée	Faible
Robustesse [8]	Élevée	Faible

3.3.2. La performance d'un essaim

La performance de l'essaim dépend considérablement de la densité des robots [177]. La performance d'un essaim sous-peuplé est près de zéro. En ajoutant de plus en plus de robots, la densité des robots continue à augmenter et la performance du système augmente aussi. En atteignant une certaine densité, la performance de l'essaim sera optimale dans la zone de cette densité. En continuant à augmenter le nombre de robots, un antagonisme résulte [193]. Le système devient surpeuplé et sa performance diminue vers zéro, à cause de l'interférence entre les robots [194].

3.3.3. La communication

Comme on l'a déjà mentionné, la communication est un aspect clé dans les systèmes de robots en essaim (SRE). Il y a deux types de communication : explicite et implicite. Dans le premier cas, des messages explicites s'échangent entre l'émetteur et le récepteur en utilisant un canal de communication. Alors que dans le deuxième cas, il n'a y pas de canaux ni de messages explicites, la communication est

[7] Le concept d'émergence est discuté dans la section 3.3.5
[8] Les trois propriétés sont discutées dans la section 3.4

faite en utilisant des indices non-intentionnels : des repères (trace) ou bien des signaux (alarme) [177]. La stigmergie est un concept pertinent de la communication implicite [177]. Les agents communiquent à travers des repères laissés dans l'environnement [175], pour ceux qui passent plus tard [177]. Ces repères peuvent être des repères chimiques tels que la phéromone déposée par les fourmis, et peuvent être aussi des modifications dans l'environnement lui-même. La stigmergie est le mécanisme par lequel un agent réagit à la modification de son environnement suite à une action d'un autre [5].

3.3.4. Les deux niveaux micro et macro

Dans un système en essaim, il existe deux niveaux [177] : (1) le niveau microscopique qui est la vue locale de l'individu avec ses capacités et connaissances limitées ; Et, (2) le niveau macroscopique qui est la vue globale de l'essaim avec une connaissance de la tâche globale. En se basant sur ces deux niveaux, les modèles de conception d'un système en essaim sont classés en modèle macro et modèle micro. Le premier modèle fait abstraction de détails microscopiques. Tandis que le modèle micro présente toutes les propriétés des agents.

3.3.5. L'émergence

Un concept philosophique vague et difficile à définir [177]. Holland [195] a écrit tout un livre sur l'émergence, mais il n'arrive pas à la définir en disant : *« Il est peu probable qu'un sujet aussi compliqué que l'émergence se soumette docilement à une définition concise, et je n'ai pas une telle définition à offrir ».* L'émergence reste une approche intéressante pour la conception des systèmes de robots en essaim robustes. C'est la clé pour obtenir un comportement global complexe à partir des comportements locaux beaucoup plus simples [196]. En d'autres termes, l'émergence nous permet d'avoir un comportement global plus grand que la somme des comportements des agents. Comme on peut le voir clairement dans les exemples naturels, l'idée c'est que les comportements individuels simples au niveau micro aboutissent, d'une manière imprévisible, à un comportement global extraordinaire au niveau macro.

3.3.6. Feedbacks

Le feedback (ou bien rétroaction) est un concept simple, mais parfois il est difficile à distinguer entre ses deux types : positif et négatif [177]. Le feedback positif c'est la force qui augmente et gonfle les quantités [177]. Prenons l'exemple de l'effet de boule de neige, le gonflage d'une montgolfière, dépôt de phéromone, d'une manière augmentée, par fourmis trouvant une source de nourriture. À certains moments cette augmentation doit être arrêtée, en utilisant un feedback négatif. Les fourmis donnent un exemple clair du feedback négatif lorsqu'elles font face aux encombrements [177, 197]. Les fourmis sur les sentiers bondés : (1) sont moins susceptible de déposer de la phéromone ; et (2) elles ont moins tendance de la déposer une autre fois lorsqu'elles le font au moins une fois [198].

3.3.7. L'auto-organisation

C'est un concept clé dans la robotique en essaim expliquant la relation entre les deux niveaux. Elle donne une réponse à la question [177] : comment des micro-interactions peuvent-ils produire une macro-structure ? L'auto-organisation est basée sur 4 composantes essentielles : feedback positif, feedback négatif, les interactions multiples et l'aléatoire [199]. Des interactions multiples sont produites par les nombreux individus qui constituent le système et qui interagissent entre eux d'une manière spatio-temporelle. Les agents continuent à répéter des comportements qui apparaissent comme étant des comportements aléatoires, l'essaim vérifie les changements (feedbacks), ensuite quelques comportements sont changés. La répétition de ces comportements aléatoires permet d'exploiter cet ensemble de comportements, afin d'apprendre dans quel sens l'exploration doit être faite.

3.4. Les propriétés d'un système de robots en essaim

3.4.1. La scalabilité

La scalabilité est définie par Bjerknes et Winfield [200] comme étant la capacité d'étendre un mécanisme auto-organisé pour un plus grand ou un plus petit nombre d'agents-robots sans influencer, considérablement, la performance du système. Le système « *peut maintenir sa fonction tout en augmentant sa taille sans qu'il soit nécessaire de redéfinir la façon dont ses parties interagissent* » [201]. Dû à l'interaction locale, le système de robots en essaim est scalable [173]. Chaque agent-robot interagit seulement avec son voisinage et il est inconscient de l'existence d'autres agents hors sa vision. La garantie d'une densité constante de robots est nécessaire pour la scalabilité de l'essaim, mais le choix de la taille de l'essaim reste libre dans cette zone où l'essaim est scalable [177].

3.4.2. La flexibilité

La flexibilité est définie comme étant la capacité du système à s'adapter aux nouvelles exigences environnementales [202], obtenue par la capacité de s'auto-organiser et le contrôle distribué [173]. En traitant la même tâche, les agents-robots peuvent être posés dans de différentes situations : la taille de l'environnement est changée, plus d'obstacles, plus ou moins d'objets à manipuler, etc. Les exigences de l'environnement peuvent être plus importantes, en changeant la tâche à traiter. L'essaim est capable de s'adapter à un large éventail de tâches parce que souvent la spécialisation de matériel n'est pas nécessaire. Les robots surmontent leurs capacités matérielles limitées par la coopération [177]. Ces robots de petite taille peuvent être auto-assemblés pour passer des trous ou bien des obstacles difficiles. Ils sont aussi capables de transporter collectivement des objets avec leurs actionneurs trop faibles [177].

3.4.3. La robustesse

La robustesse est la tolérance aux fautes (aux pannes) [177] ; c'est la façon dont le système se débrouille même si quelques robots ne réussissent pas à poursuivre leur mission [173]. La redondance, le contrôle distribué et l'interaction locale rendent le système hautement tolérant aux fautes. Les agents-robots restants peuvent compenser l'absence *partielle* de leurs collègues grâce à la redondance (homogénéité) [177]. À cause de l'interaction locale et le contrôle distribué, la perte des agents-robots n'a qu'un effet local [177] et n'entraine pas une défaillance globale du système [173] ; même si l'efficacité du système se dégrade en cas de perte d'un nombre important d'agents [177]. En utilisant un contrôle distribué, aucune réallocation ou ré-coordination n'est nécessaire

3.5. Taxonomie de la robotique en essaim

Nedjah et Junior [50], il n'y a pas une taxonomie consensuelle, vu l'évolution continue et constante de la robotique en essaim. Plusieurs taxonomies ont été proposées, parmi elles, celles proposées par : Bayindir et Şahin [203], Brambilla et al. [204], Cao et al. [205] et Iocchi et al. [206]. Conformément à la taxonomie de Brambilla et al. [204], la plus concise [50], les travaux de recherche peuvent être classifiés selon les méthodes et/ou les comportements collectifs (voir la figure 3.1). Les méthodes sont divisées en deux catégories : les méthodes de conception et les méthodes d'analyse. Et sous les comportements collectifs, on trouve : les comportements d'organisation spatiale, les comportements de navigation, les comportements de prise de décision collective et d'autres.

La conception pourrait être faite d'une manière **manuelle** (appelée aussi conception basée sur le comportement). Elle est fondue sur les principes observés chez les organismes naturels suivant un processus d'essai-erreur [201, 204]. La conception est faite par *une machine probabiliste à états finis* où la transition entre les états est faite en fonction des probabilités. Ou bien en utilisant une conception basée sur *la physique virtuelle*, où chaque robot exerce des forces virtuelles sur les autres. Par exemple,

une force d'attraction, de répulsion et d'orientation dans le comportement de mouvement coordonné, présenté dans la section 3.6.6. D'autres techniques peuvent être utilisées pour une conception manuelle telle que : la définition d'un ensemble de propriétés du système désiré [207]. La limite principale de la conception manuelle c'est qu'elle est liée à l'expérience et l'ingéniosité du concepteur humain [201].

Les méthodes de conception **automatique** génèrent les comportements collectifs désirés sans intervention explicite du concepteur [204]. Elles regroupent la robotique évolutionnaire (RE), l'apprentissage par renforcement et d'autres méthodes. *La robotique évolutionnaire*, définie comme étant l'application d'une méthode évolutionnaire (discutée dans le chapitre 1) pour concevoir un système mono ou multi-robot [50, 204], reste la plus souvent utilisée. Les robots de l'essaim apprennent génération après génération comment accomplir la tâche accordée [177]. L'une des limitations de la RE est la difficulté de trouver le paramétrage évolutionnaire adéquat [201]. Trianni [171] a fait une synthèse des travaux les plus intéressants sur la conception de comportements collectifs auto-organisés en exploitant les techniques de la RE.

Si le contrôleur est conçu en appliquant la méthode évolutionnaire dans une phase initiale de conception, ensuite ce contrôleur est utilisé dans la phase de fonctionnement. Ce processus est appelé *hors-ligne.* Tandis que dans le cas *en ligne*, le contrôleur adéquat est sujet de recherche pendant le fonctionnement [50]. Le coût de calcul élevé des algorithmes évolutionnaires rend leur utilisation en ligne peu pratique surtout en robotique mobile, qui n'offre pas de solution à ce genre de problèmes [50]. Pour cette raison, la robotique évolutionnaire hors-ligne est plus populaire que celle en ligne.

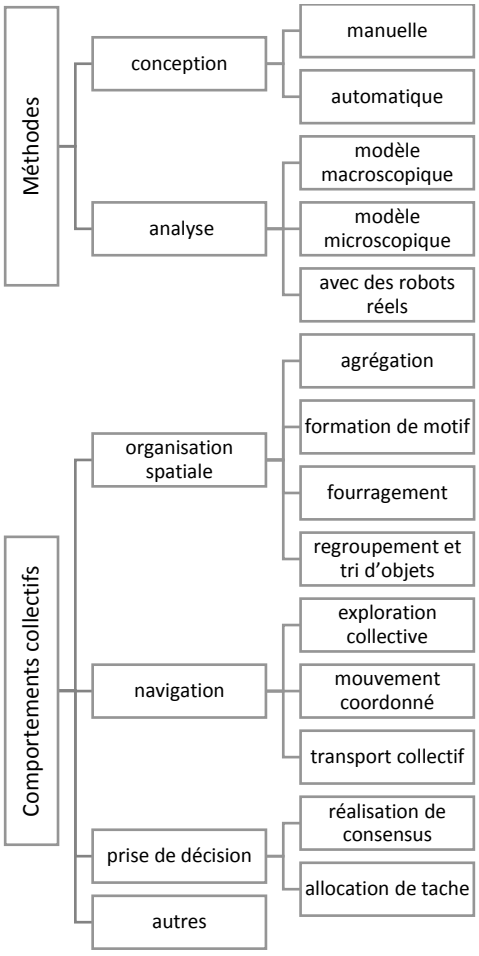


Figure 3. 3 : Taxonomie proposée par Brambilla et al. [204]

Après avoir conçu le système, une analyse doit être faite, généralement en utilisant des modèles, dans le but de voir si l'essaim présente le comportement collectif désiré avec les caractéristiques désirées [204]. Il existe deux modèles : le modèle macroscopique et le modèle microscopique. Le système est généralement modélisé en utilisant l'un de ces deux modèles, mais pas les deux à la fois à cause de la difficulté trouvée vu la nature de ces systèmes auto-organisés [204].

Dans un **modèle macroscopique**, seulement les caractéristiques de l'essaim sont décrites et les individus ne sont pas pris en compte. Sous cette classe, il existe des travaux faisant appel aux *équations de taux*. Ces équations décrivent l'évolution temporelle de la proportion de robots dans un état donné au nombre total de robots. *Des équations différentielles* empruntées de la physique statistique sont aussi utilisées pour modéliser le système. En plus de ces deux exemples, d'autres approches sont utilisées [204].

Par contre dans le **modèle microscopique**, les comportements individuels des robots et leurs interactions sont explicitement modélisés par le biais de simulation, avec 3 niveaux d'abstraction [204]. La simulation la plus simple considérant les robots comme des points matériels, la simulation 2D, et la simulation 3D avec plus de détails. La simulation reste la méthode la plus utilisée pour analyser et valider le système en essaim conçu. Dans la section 3.9, on mentionne quelques simulateurs utilisés dans ce cadre.

L'analyse est aussi faite **en utilisant des robots réels**. Ce qui est important pour distinguer entre les comportements réalisables et irréalisables dans la pratique. Les expériences sont faites sous des contraintes de laboratoire telles que la douceur de sol, la luminosité, les interférences radio, etc. Et pour cette raison elles restent loin de prouver que l'essaim va fonctionner dans des applications réelles. Dans la section 3.9, les robots physiques les plus utilisés et/ou les plus récents sont présentés.

Les comportements collectifs (appelés aussi scénarios [177] ou bien problèmes) étudiés dans la robotique en essaim peuvent être : des comportements d'**organisation spatiale** qui se concentrent sur la façon dont les robots ou les objets sont organisés et distribués dans l'environnement [201]. Parmi les comportements d'organisation spatiale de robots : *l'agrégation*, *la formation de motif* et *la formation de chaine* (connexion logique entre les robots), *l'autoassemblage et la morphogénèse* (connexion physique). Et les comportements d'organisation spatiale d'objets sont : *le fourragement*, *le regroupement, le tri et le tri en anneaux*, discutés dans la section suivante.

Les scénarios de **navigation** se focalisent sur la manière dont les mouvements de robots sont organisés et cordonnés [201]. Tels que *l'exploration collective* (les robots coopèrent pour explorer l'environnement), *le mouvement coordonné* (les robots bougent ensemble) et *le transport collectif* (les robots transportent ensemble un objet trop lourd pour un seul robot).

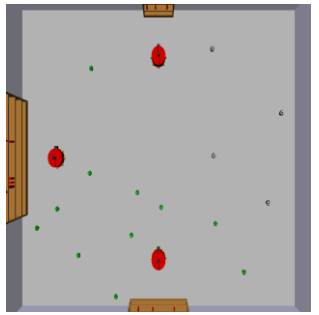
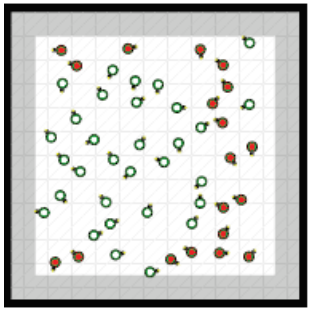
La classe de **prise de décision** collective vise à permettre à l'essaim de se mettre d'accord [204] pour *une réalisation consensuelle* (prendre une décision conjointe) ou bien pour *une allocation de tâches* (de se répartir entre plusieurs tâches). L'exemple de prise de décision dans les essaims naturels est celui de chois de site de nidification par les abeilles éclaireuses. Elles s'influencent mutuellement pour prendre un choix adéquat.

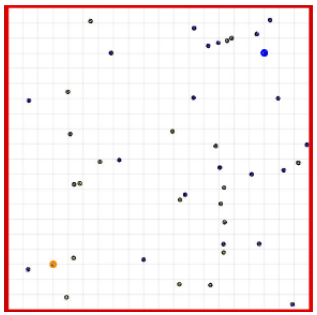
Et d'**autres** scénarios, qui n'appartiennent à aucune classe déjà mentionnée, tels que *la détection collective de faute*, *l'interaction humain-essaim* et *la régularisation de taille* de l'essaim [204].

Quelques comportements collectifs représentatifs de chaque classe sont décrits dans la section suivante.

3.6. Les scénarios de la robotique en essaim

Tableau 3.3 : Les scénarios étudiés dans la robotique en essaim

Scénario	Description	Source d'inspiration	Les travaux	Exemple
3.6.1. Agrégation **(Aggregation)**	**Le principe :** Les robots ont comme tâche de se positionner l'un près de l'autre dans un seul endroit [177]. **L'intérêt** : Elle représente une condition initiale et essentielle pour d'autres tâches de coopération [171], telles que la formation de motifs.	Une agrégation auto-organisée est observée chez plusieurs organismes naturels tels que les bactéries, les mammifères [211], les papillons monarques [177], les blattes [212], etc.	[208] [211] [212] [213] [214]	 **[208]**
3.6.2. Allocation de tâche **(Task allocation)**	**Le principe :** Les robots se répartissent entre les tâches. Chaque robot choisit sa tâche dynamiquement, pour augmenter la performance du système [204]. **L'intérêt :** L'allocation de tâches est très demandée pour les processus de production, les tâches complexes, etc.	Ce comportement est observé par exemple chez les fourmis et les abeilles qui se partagent des tâches comme le fourragement et la surveille de larves [204].	[209] [215] [216] [217]	 **[209]**
3.6.3. Détection collective de faute **(Collective fault detection)**	**Le principe :** Les robots sont capables de détecter si les robots dans leur voisinage ont une panne ou non. **L'intérêt :** Il est très important que l'essaim détecte les défaillances dans toutes les tâches accordées.	Inspiré des éclats synchronisés chez les lucioles. Si un individu n'est pas synchronisé avec les autres, il est supposé d'être en panne [210].	[210] [218] [219]	 **[210]**

Scénario	Description	Source d'inspiration	Les travaux	Exemple
3.6.4. Exploration collective **(Collective exploration)**	**Le principe :** Les robots coopèrent pour explorer un environnement inconnu. **L'intérêt** : L'exploration représente la tâche fondamentale pour d'autres telles que l'affectation, trouver le plus court chemin pour une tâche de fourragement, etc.	Observée chez les fourmis qui trouvent le plus court chemin en utilisant la phéromone, les abeilles par le biais de dance.	[220] [223] [224]	 [220]
3.6.5. Fourragement **(Foraging)**	**Le principe :** Les robots ramassent des objets dispersés dans l'environnement et les déposent dans un endroit spécifié appelé maison [171]. **L'intérêt :** Plusieurs applications pratiques telles que : le déminage [191], recherche et récupération [177], etc.	En utilisant des interactions locales seulement, les colonies de fourmis, les abeilles, et d'autres organismes naturels peuvent exploiter des sources de nourritures [191].	[221] [225] [226] [227]	 [221]
3.6.6. Mouvement coordonné[9] **(Coordinated motion)**	**Le principe :** Les robots ont comme tâche la coordination de leurs mouvements afin de se déplacer d'une manière cohérente [171], dans la même direction et avec la même vitesse [177]. **L'intérêt :** Bénéfique pour les tâches d'exploration simultanée sensible aux pertes d'information.	Inspiré de bancs de poissons ou bien le flockage d'étourneaux [171](voir tableau 3.1). Observé aussi chez les criquets [177].	[222] [228] [229]	 [222]

[9] Appelé aussi mouvement collectif (collective motion) ou bien flockage (flocking) [177].

Scénario	Description	Source d'inspiration	Les travaux	Exemple
3.6.7. Regroupement d'objets[10] **(Object clustering)**	**Le principe :** Les robots ont comme tâche la collection, dans un seul tas, un ensemble d'objets distribués dans l'environnement. **L'intérêt** : potentiellement utilisé dans des tâches de nettoyage.	Inspiré d'un comportement de formation d'un cimetière observé chez les fourmis. Les cadavres sont regroupés dans un seul tas en attirant les ouvriers à déposer d'autres.	[196] [6] [230] [4] [231]	[6]
3.6.8. Tri d'objets **(Patch/Object sorting)**	**Le principe :** Les robots classifient des objets selon leurs caractéristiques telles que la couleur, la forme, etc. Les clusters doivent être complètement séparés. **L'intérêt :** Utile pour la classification de produits, l'isolation des objets dangereux, etc.	Inspiré du tri de couvée observé chez les fourmis. Elles construisent trois tas séparés d'œufs, de larves, et de pupes.	[6] [232] [233] [234]	[6]
3.6.9. Tri en anneaux[11] **(Annular sorting)**	**Le principe :** Un groupe central du même type d'objets entouré par des anneaux construits par des objets du même type [177]. **L'intérêt :** Utile dans des processus de construction, d'encapsulation et les stratégies de défense [236].	Inspiré du tri de couvée *o*bservé chez les fourmis, lors le déménagement. Placer les œufs au centre ensuite les micro-larves, les larves (dans un ordre croissant), les prépupes, ensuite les pupes [6].	[6] [235] [236] [237]	[6]

[10] Ce scénario est celui auquel on s'intéresse dans cette thèse, il est discuté en détail dans la section 3.8
[11] Connu aussi sous le nom ségrégation (segregation).

3.7. Le problème d'énergie

Dans la section de robustesse 3.4.3, on a parlé de la défaillance, mais quelles sont les causes ? La défaillance des robots peut être une défaillance partielle des capteurs, des moteurs ou bien une défaillance totale de robot [200]. La dernière est généralement causée par la durée de vie limitée des batteries. Avec la particularité que les robots de l'essaim sont des robots simples dotés d'une batterie avec une capacité beaucoup moins limitée que d'autres robots plus complexes ; la durée de vie représente la principale contrainte non seulement pour l'autonomie de l'essaim, mais pose également de sérieuses contraintes opérationnelles, au point de compromettre leur utilisation pour des tâches complexes.

3.7.1. Solutions proposées dans la robotique

Récemment, davantage attention est accordée à la conception de systèmes robotiques prenant en compte la minimisation de l'énergie consommée. Les méthodes et les techniques, proposées dans le cadre de résolution du problème d'énergie dans la robotique en général, sont classées selon [238] en *matérielles*, *logicielles et mixtes*.

La première catégorie, **matérielle**, exploite les nouveaux matériaux pour la conception des composants plus légers et des batteries plus puissantes. L'utilisation de composants plus légers, par exemple le polymère ultraléger renforcé de fibres [239], va réduire le poids, ce qui facilite le contrôle et minimise la consommation d'énergie, en conséquence. Aussi des améliorations sont faites dans ce sens sur la pression de l'air et le diamètre des bras, la conception et le positionnement des moteurs, etc. [238]. Cette catégorie inclut aussi des stratégies de récupération de l'énergie en réutilisant l'air sous pression d'actionneurs au lieu de le libérer, et des stratégies de distribution d'énergie entre les actionneurs. L'énergie de freinage par exemple est transférée aux autres actionneurs qui accélèrent ou bien elle est stockée [238, 240].

Dans la deuxième catégorie, **logicielle**, les paramètres de système (tels que : les masses, les paramètres électriques d'actionneurs) sont passés comme des entrées à un modèle de simulation pour analyser les différentes stratégies. L'impact de ces paramètres sur la consommation d'énergie est évalué. Les approches de cette catégorie reposent sur l'optimisation de la phase de planification de mouvements pour réduire le temps et le trajet, ce qui va réduire l'énergie consommée en conséquence [238]. Les systèmes étudiés généralement dans ce sens sont des robots manipulateurs avec plusieurs degrés de liberté. Aussi la planification des opérations, d'une ligne de robots qui travaillent d'une manière synchrone, est sujette d'optimisation. La réduction du temps d'inactivité de robots, qui attendent leur tour, va réduire la consommation de l'énergie [238, 241].

La troisième catégorie, **mixte**, regroupe les travaux faits dans les deux sens en même temps, matériel et logiciel. Barreto et al. [242] ont modifié le système physique en ajoutant des éléments élastiques au mécanisme, avec une planification adéquate des mouvements. De cette manière, une grande partie de l'énergie nécessaire pourrait être fournie par ces éléments. Ils stockent l'énergie comme étant une énergie élastique potentielle et la libèrent vers les éléments rigides pour augmenter leur énergie cinétique [238]. Aussi des dispositifs de partage d'énergie sont ajoutés aux systèmes à plusieurs degrés de liberté ou bien les systèmes multi-robots, avec de nouveaux scénarios de planification de mouvements, afin de récupérer et partager l'énergie gaspillée (telle que l'énergie de freinage) [243].

3.7.2. Solutions proposées dans la robotique en essaim

On distingue 3 différentes classes de solutions proposées pour traiter le problème d'énergie, dans la robotique en essaim : le chargement d'énergie, l'homéostasie énergétique collective et la minimisation de l'énergie consommée. Et à leur tour ces 3 classes sont divisées en sous-classes présentées sur la Figure 3.2. Dans ce qui suit, on mentionne un exemple de travaux faits dans chaque sous-classe.

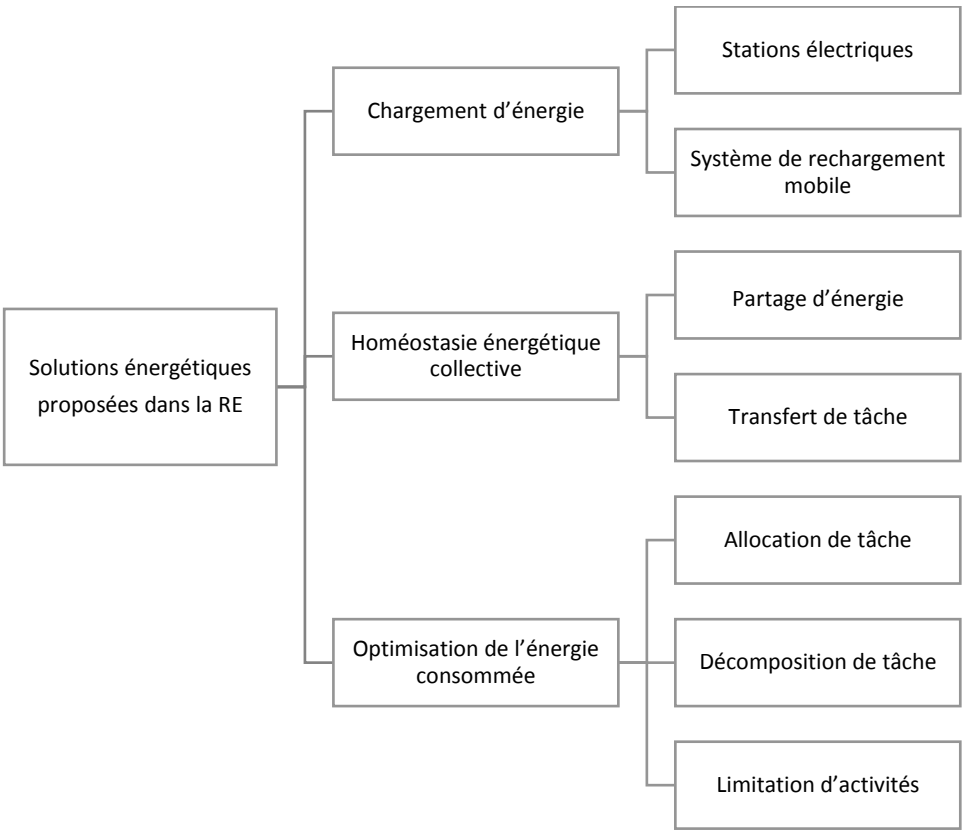


Figure 3.1 : Classification de solutions proposées au problème d'énergie dans la RE

A. Chargement d'énergie :

A.1. Stations électriques : Le mécanisme de chargement d'énergie avec des stations électriques a été examiné par Al Haek et al. [202]. En cas de faible énergie, les robots doivent trouver la station électrique la plus proche, dans l'environnement où ils performent une tâche de recherche de nourriture (fourragement). Parmi les limites et les lacunes de l'utilisation des stations électriques, indiquées par Al Haek, et al. [202] : Il n'est pas possible de placer des stations électriques dans l'environnement d'exécution d'une tâche, dans tous les cas ; et les robots doivent avoir des connaissances sur l'environnement, ce qui est en contradiction avec certains principes de robotique en essaim.

A.2. Système de rechargement mobile : Carrillo et al. [223] ont présenté un scénario d'exploration collaborative en utilisant un système de rechargement d'énergie mobile, avec deux types de robots mobiles : les robots d'exploration et les robots de recharge de batteries. Le problème est traité comme un problème à deux objectifs : (1) maximiser la zone explorée par les robots ; et (2) minimiser le risque de panne de batterie dans l'essaim. L'algorithme évolutif multi-objectif NSGA-II [32] est utilisé pour résoudre ce problème. Les résultats montrent qu'avec un petit rayon de détection, le gain de l'utilisation des fonctionnalités de rechargement est plus important, car la consommation d'énergie est plus élevée, dans ce cas.

B. Homéostasie énergétique collective :

B.1. Partage d'énergie : La réponse du système immunitaire, appelée formation de granulomes, a été la source d'inspiration d'une stratégie de partage de l'énergie proposée par Ismail, et al. [244]. Un robot qui souffre d'un manque d'énergie est considéré comme une cellule infectée. D'autres robots entourent ce robot et partagent avec lui une partie de leur énergie, après avoir reçu le signal d'aide. Cette approche reste efficace si la moyenne de l'énergie de tous les robots est suffisante pour accomplir la tâche. Mais si ce n'est pas le cas, partager l'énergie signifie que l'énergie des robots immunitaires diminuera considérablement, surtout si le nombre de robots infectés est important. Et à un moment donné, tout le système va s'effondrer.

B.2. Transfert de tâche : Zhou et Kinny [245] ont proposé un algorithme d'optimisation par essaim de particules à base d'énergie (EPSO pour Energy-based Particle Swarm Optimization). Les robots ayant un bon niveau d'énergie peuvent aider leurs voisins à accomplir leur tâche, au lieu de transférer l'énergie à ces robots incapables énergétiquement de l'accomplir tous seuls. Le cas d'étude dans ce travail est un scénario de fourragement. Les robots poussent des objets distribués afin de les transporter vers leur dépôt. Après avoir trouvé un objet, le robot décide s'il peut pousser cet objet. Dans le cas où son énergie est insuffisante, il demande de l'aide. De cette manière, la variance d'énergie entre les robots est réduite et les robots peuvent pousser plus d'objets.

C. La minimisation de l'énergie consommée :

C.1. Allocation de tâche : Lee et al. [246] ont présenté un autre modèle inspiré d'essaim d'abeilles mellifères, pour améliorer l'efficacité énergétique dans la tâche de fourragement. Les robots sont divisés en fourrageurs, spectateurs et éclaireurs. Où les éclaireurs se déplacent au hasard et recueillent de l'information sur les aliments, comme la quantité, la distance et l'emplacement. Les spectateurs reçoivent ces informations et sur la base de celles-ci, ils donnent un ordre aux fourragers. Les résultats montrent l'efficacité du modèle en termes d'énergie consommée.

C.2. Décomposition de tâche : Lee et Ahn [174] ont proposé un modèle comportemental qui minimise l'énergie consommée. La tâche de fourragement est divisée en deux sous tâches : la collection d'objets dans des caches potentiels[12] et la transportation de ces objets vers la base. L'espace de recherche est divisé en régions constantes selon le nombre de caches et les objets trouvés dans une région sont stockés dans le cache dédié à cette région. Par la suite, les robots transportent ces objets vers la base. Les auteurs ont prouvé que leur modèle surpasse le modèle conventionnel en termes de consommation d'énergie. Cette approche réduit la recherche répétitive autour de la base à chaque fois qu'un robot dépose un objet, ce qui permet de conserver leur énergie.

C.3. Limitation d'activités : Un mécanisme d'optimisation par essaim de particules conscient de l'énergie (EAPSO pour Energy Aware Particle Swarm Optimization) a été proposé par Mostaghim, et al. [247]. Des microrobots aériens ayant une capacité énergétique limitée ont été utilisés pour accomplir une tâche d'exploration. Chaque robot décide si son énergie est suffisante pour voler ou s'il doit rester au sol. Avec cette condition le problème devient bi-objectif : 1) le profit en termes de gain global dans le processus de recherche ; et 2) le coût en termes de consommation énergétique. La somme pondérée (discutée dans le chapitre 1) a été utilisée pour transformer le problème en un problème mono-objectif.

[12] Des endroits de dépôt temporaire.

3.8. Le regroupement d'objets

3.8.1. Source d'inspiration

Le scénario de regroupement d'objets est inspiré d'un comportement observé chez les fourmis [5]; qui présentent une variété de comportements collectifs connus comme les plus matures parmi les comportements inspirés de la nature [230]. Le phénomène de formation d'un cimetière est observé chez plusieurs espèces de fourmis [5, 6, 7]. Sans avoir transporté les fourmis mortes et les déchets loin du nid, la moisissure va se développer dans le nid [7]. Dans quelques heures, les ouvriers peuvent rassembler des cadavres distribués aléatoirement. Les clusters de cadavres grossissent en attirant les ouvriers à déposer d'autres cadavres [230], à travers le mécanisme de feedback positif [6]. Les clusters qui contiennent plus de cadavres sont plus susceptibles d'évoluer et les petits clusters sont plus susceptibles de disparaitre [230]. Dans ce cas, la fourmi ramasse ou dépose un cadavre, d'une manière probabiliste, selon son estimation partielle de la taille du cluster. Martin et al. [5] ont prouvé que des fourmis virtuelles beaucoup plus simples, sans avoir la possibilité d'utiliser les probabilités de ramassage/dépôt, font exactement le même travail, même si ce manque d'intelligence ralentit le processus.

3.8.2. Définition et types de scénarios

Le regroupement d'objets, appelé aussi formation du tas [4] est la tâche dont un cluster final émerge de l'assemblage et du désassemblage continus des sous-clusters. Contrairement à la tâche de fourragement, l'emplacement de collection d'objets n'est pas prédéfini. Les robots doivent décider où le tas sera formé en utilisant une approche déterministe ou probabiliste [191].

Dans un scénario **déterministe**, les robots ont des capacités de localisation et peuvent échanger des informations sur l'emplacement de leurs clusters actuels. Au début chaque robot choisit un emplacement aléatoire. Au fur et à mesure, les robots s'influencent mutuellement pour se mettre d'accord sur l'emplacement final. Ce qui est similaire à la tâche de prise de décision conjointe. Dans ce cas, la communication, directe ou indirecte, entre les robots joue un rôle important dans l'émergence du cluster.

D'autre part, dans le scénario **probabiliste** qui nous intéresse, les robots ont une représentation limitée de l'environnement, sans mémoire ni capacité de localisation, et il n'y a pas de communication inter-robots. La décision de savoir quand et où se déplacer, ramasser ou déposer des objets est prise en fonction d'une perception locale individuelle. Donc, le regroupement d'objets probabiliste peut être défini comme le résultat d'une quantité importante de hasard et d'interactions purement locales, sans transfert global des informations ou contrôle central. Dans ce qui suit, on présente quelques scénarios probabilistes étudiés dans la littérature.

3.8.3. Quelques travaux faits sur le regroupement d'objets probabiliste

Hartmann [6] présente un scénario dans lequel des agents doivent effectuer une tâche de regroupement dans une grille de cellules carrées (voir l'exemple de regroupement d'objets présenté dans le tableau 3.3). Ces agents ont une portée de détection de 8 cellules voisines, avec les 6 comportements suivants : avancer, reculer, tourner à gauche, tourner à droite, ramasser ou déposer l'objet. Pour mettre au point la meilleure solution, Hartmann [6] utilise un algorithme génétique pour 6000 générations avec une population de 50 individus et 4000 pas de temps pour chaque cycle d'entraînement.

Dans le modèle proposé par Chatty et al. [230], chaque agent fait une recherche aléatoire pour trouver les objets. Après avoir trouvé des objets dans son voisinage, il choisit l'objet le plus proche pour le prendre s'il est dans le mode prise, et il choisit l'objet le plus loin pour déposer à côté de lui s'il est dans le mode dépôt. En se basant seulement sur ce mécanisme, et sans passer par une étape d'entrainement,

les agents-robots réussissent à former le tas, mais le nombre de prise/dépôt est beaucoup plus important par rapport à l'approche basée sur l'entrainement (hors-ligne).

Alors que Barfoot et D'eleuterio [4] ont développé un modèle de 60 objets à regrouper par 30 agents-robots qui ont une perception locale de 5 cellules voisines et 2 comportements : se déplacent et manipulent un objet. Une fonction d'entropie de Shannon modifiée [4] est utilisée comme métrique pour évaluer les solutions obtenues par l'algorithme génétique. Les auteurs suggèrent de diviser l'environnement en 9 parties et la meilleure solution est celle où tous les objets sont rassemblés dans une seule partie.

En outre, Vorobyev et al. [196] ont utilisé le nombre d'objets dans deux zones de détection différentes, la zone centrale courte et la zone d'exploration longue, comme entrées d'un réseau neural qui donne en sortie un des trois comportements suivants : faites demi-tour, tournez et bougez tout droit. De plus, les robots sont équipés d'une pelle frontale qui permet de pousser de nombreux objets à la fois. La taille du plus grand cluster est utilisée comme métrique pour évaluer les solutions pendant le processus d'apprentissage supervisé proposé.

Par le biais d'expériences faites par des robots physiques, Gauci et al. [231] ont prouvé qu'une tâche de regroupement peut être réalisée en utilisant des robots très simples : sans mémoire, sans calcul, équipés de seulement un capteur frontal qui perçoit un objet ou un robot à la fois et sans pinces. En se basant sur la perception actuelle, le contrôleur choisit la vitesse angulaire des roues pour éviter un autre robot, pousser l'objet perçu ou continuer à se déplacer. Afin de trouver le meilleur contrôleur, l'algorithme évolutif CMA-ES (Covariance Matrix Adaptation Evolution Strategy) est utilisé et la dispersion des objets est mesurée [231].

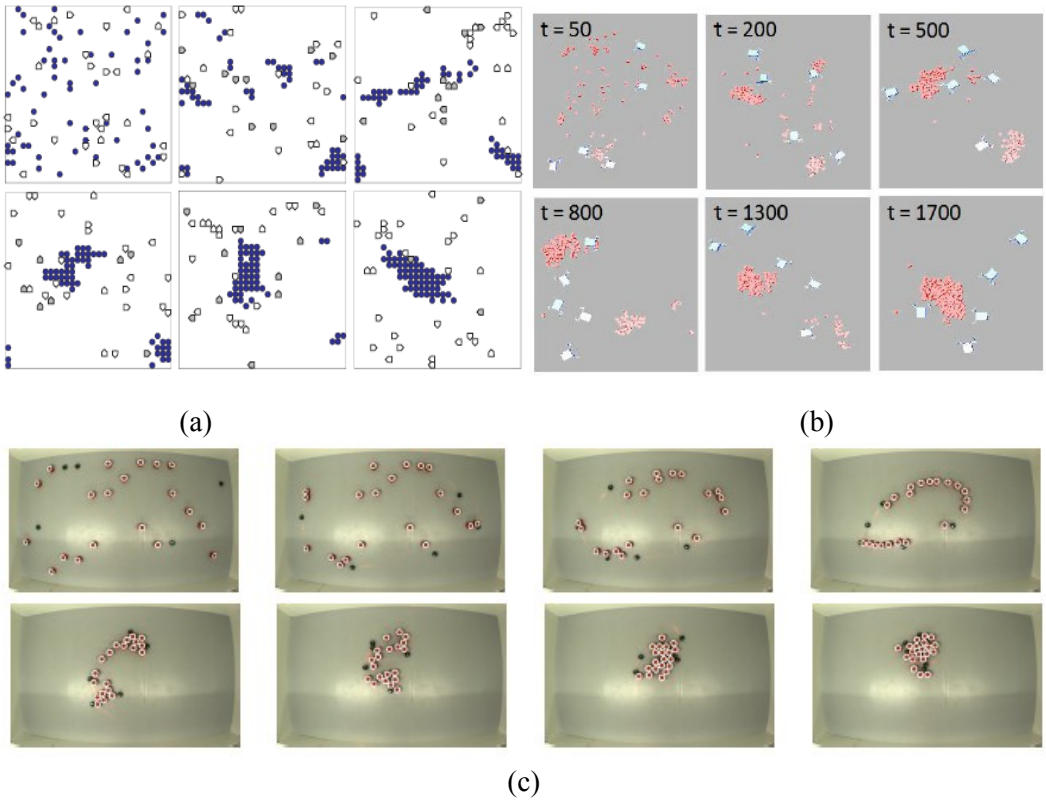
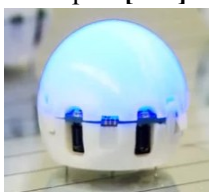
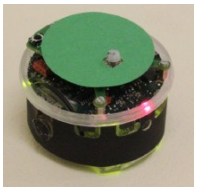


Figure 3.2 : Le scénario de : (a) Barfoot et D'eleuterio [4], (b) Vorobyev et al. [196] et (c) Gauci et al. [231].

3.9. Simulateurs et robots physiques

Les méthodes ou les comportements décrits dans la phase de conception logicielle peuvent être validés en utilisant la simulation ou bien des plateformes de robots physiques. Le tableau 3.4 résume les plateformes de robots les plus récentes et/ou les plus utilisées dans des tâches de robotique en essaim. Il faut noter qu'ils ne sont pas conçus pour former des essaims seulement, mais ils peuvent être utilisés individuellement [50].

Tableau 3.4 : Robots en essaim physiques

Prototype	Spécifications	Utilisé dans
Alice **[248]**  Prix : /	*Taille* :2.2cm *Capteur(s)* : distance, caméra *Mouvement* : roues *Vitesse* : 4cm/s *Autonomie* : jusqu'à 10h	- Fourragement [227] - Couverture coordonnée (dispersion) [249]
AMiR **[250]**  Prix : 65£	*Taille* : 6.5cm *Capteur(s)* : distance, lumière, orientation *Mouvement* : roues *Vitesse* : 10cm/s *Autonomie* : 2h	- Agrégation [213, 214]
Droplet **[251]**  Prix:/	Taille :4.4cm Capteur(s) : lumière, portée, orientation, couleurs RVB Mouvement : pattes/vibration Vitesse : / Autonomie : +24h	- Camouflage distribué [252] - Le projet vise à créer un « liquide qui pense » [253]
E-puck **[254]**  Prix :580£	*Taille* : 7.5cm *Capteur(s)* :IR, distance, caméra, accéléromètre, microphone *Mouvement* : roues *Vitesse* : 13cm/s *Autonomie* : jusqu'à 10h	- Agrégation [208] - Fourragement [221] - Tri en anneaux [237]
Kilobot **[255]**  Prix : 12£	*Taille* : 3.3cm *Capteur(s)* : distance, lumière *Mouvement* : vibration *Vitesse* : 1cm/s *Autonomie* :3-24h	- Auto-assemblage [256] - Prise de décision collective [257] - Transport collectif [258]
Autres	Cellulo [259], Colias-Φ [260], Khepera IV [261], Mona [262], R-one [263]	

Tandis que, les simulateurs utilisés ne sont pas dédiés spécialement à la robotique en essaim. Ils varient des [264] simulateurs orientés robots tels que : ARGoS [265], Enki [266], Roborobo [267], V-REP [268] et Webots [269] aux simulateurs orientés agents basés sur les principes de système multi-agent tels que NetLogo. Les systèmes multi-agents représente un paradigme de calcul distribué [270, 271] utilisé pour modéliser et contrôler les systèmes complexes [272] tel que les systèmes multi-robot. Selon la relation entre les perceptions et les actions, on distingue deux types d'architecture de ces entités software, appelées agents : les agents cognitifs et les agents réactifs. Dont les agents cognitifs effectuent une certaine délibération pour agir. Alors que les agents réactifs ne font qu'acquérir les perceptions et appliquer certaines règles prédéfinies.

3.10. Domaines d'application potentiels et limitations

L'utilisation de la robotique en essaim est bénéfique dans 4 cas [176, 273, 192] : (1) les tâches qui nécessitent un travail parallèle, pour couvrir une grande zone, afin de gagner de temps. (2) Les tâches dangereuses pour un robot cher telles que l'aide après les catastrophes, les tâches de minage/déminage et d'autres tâches militaires. (3) Les tâches qui demandent une population scalable et robuste, où les individus peuvent joindre/quitter l'essaim sans causer des interruptions, ce qu'est très utile dans plusieurs tâches réelles dans des environnements dangereux. (4) Et les tâches qui exigent une conception économique telle que le processus de fourragement dans le domaine d'agriculture. Et dans certaines tâches où les robots sont irrécupérables, l'utilisation des robots coûteux est économiquement inacceptable. Bien que la conception d'un système en essaim soit moins chère qu'un seul robot monolithique sophistiqué, il est réutilisable dans plusieurs tâches, en apportant des changements logiciels mineurs.

Bien que les études faites sur la robotique en essaim soient très intéressantes et les propriétés prometteuses qui prouvent que l'utilisation d'un système de robots en essaim sera très bénéfique, voir indispensable dans certains cas, leur application commerciale réelle est, malheureusement, limitée à cause de [274] : (1) la caractéristique principale d'émergence : il n'existe pas, pour le moment, une méthode générale de conception permettant un passage direct entre les comportements de robots et le comportement global ; (2) l'indisponibilité d'une bonne infrastructure de laboratoires pour pouvoir réaliser plusieurs études théoriques et simulations intéressantes ; (3) la difficulté de construire un modèle mathématique d'un système en essaim, ce qui pourrait être nécessaire pour développer une application valide et optimale. Nedjah et Junior [50] ont résumé ces 3 points comme étant un manque de standardisation matérielle et logicielle.

3.11. Conclusion

La robotique en essaim est un domaine multidisciplinaire, difficile à étudier [50], qui a reçu beaucoup d'attention ces dernières années. Les chercheurs prévoient que dans le futur proche, les systèmes en essaim seront de plus en plus utilisés dans le monde réel [204]. Puisqu'elle vise à développer des systèmes robustes, scalables et flexibles, elle sera très bénéfique pour des applications réelles et commerciales, une fois ce domaine surmonte ses limitations actuelles. On vous propose ces enjeux suspendus comme étant des orientations de recherches, pour pousser la robotique en essaim vers la standardisation et l'application réelle : (1) adopter/développer un langage formel pour la spécification des exigences comme une première étape ; (2) développer une plateforme de test qui inclut des métriques et des benchmarks standards ; (3) étudier l'interaction humain-essaim.

Dans le chapitre suivant, on va décrire notre proposition pour synthétiser un ensemble de contrôleurs (modèles comportementaux) qui prennent en considération la qualité de regroupement et l'énergie consommée.

Chapitre 4: Un modèle comportemental optimisant la consommation d'énergie et la qualité de regroupement d'objets

4.1. Introduction

Jusqu'à présent, la tâche de regroupement d'objets était traitée comme un problème à objectif unique ; le seul sujet de recherche était un modèle comportemental qui réussissait à former un tas final avec une qualité maximale. Cependant, lorsqu'elle doit être effectuée en continu, sans possibilité de recharger les batteries à proximité, la consommation d'énergie devient un véritable problème et le choix d'actions consommant moins d'énergie devient une nécessité pour accomplir correctement la tâche. Sinon, la mission peut être interrompue en raison du nombre croissant de robots hors service. C'est pourquoi ce problème a été remodelé pour le résoudre comme un problème d'optimisation bi-objectif. Pour ce faire, il faut trouver un modèle comportemental permettant au système robotique de minimiser la consommation d'énergie et de maximiser la qualité de formation des tas. Pour répondre à la question comment concevoir les comportements des agents-robots pour que le comportement global émerge, plusieurs approches ont été proposées. Les algorithmes évolutionnaires sont les plus populaires [196].

Le reste de ce chapitre est organisé comme suit. La section 4.2 décrit la motivation du travail présenté. Dans la section 4.3, le présent travail est classifié selon la taxonomie décrite dans le chapitre précèdent. Le problème de regroupement d'objets bi-objectif est décrit au niveau macro dans la section 4.4, en présentant les deux fitness de qualité et de consommation d'énergie. Ensuite, le modèle proposé est présenté dans la section 4.5 en décrivant le module de perception, les comportements de base et le module de contrôle. Tandis que, la section 4.6 présente la structure des agents-robots utilisés. Et la section 4.7 décrit l'algorithme évolutionnaire NSGA-II utilisé pour résoudre le problème. Les résultats obtenus sont présentés dans la section 4.8.

4.2. Motivation

Comme on l'a déjà vu dans la section 3.8.3, les travaux proposés dans le cadre de la résolution du problème de regroupement d'objets probabiliste sont variés en termes de : (1) Scénarios possibles où le nombre de robots, le nombre d'objets, la taille de l'environnement, la taille de la région de perception locale et les comportements de base possibles sont variés ; (2) Métriques utilisées pour mesurer la performance du système conçu dépondant de la fonction d'entropie de Shannon [4], la dispersion des objets [231], la taille du plus grand cluster [196] et d'autres ; (3) Méthodes de conception qui peuvent être manuelles comme dans [230], automatiques en utilisant les algorithmes évolutionnaires [6, 4, 231] ou un apprentissage supervisé [196] ; (4) Méthode d'analyse (ou bien de modélisation) faite en utilisant un modèle microscopique dans [6, 230, 4, 196] ou bien en utilisant des robots réels et la simulation [231].

Quoique les travaux déjà réalisés sur le regroupement d'objets sont variés en termes de scénarios possibles, métriques utilisées ou méthodes de conception et d'analyse. Le point commun entre ces travaux c'est que l'objectif unique de résolution de ce problème était toujours l'optimisation de la qualité du tas formé. Alors que si le système doit opérer d'une manière ininterrompue dans un environnement sans possibilité de rechargement, la minimisation de l'énergie consommée doit être prise en considération afin d'accomplir la tâche. Parmi les lacunes de conception d'un système de robots en essaim qui ne prend pas en considération l'énergie consommée lors de l'accomplissement de la tâche [9] nous retrouvons :

(1) L'énergie consommée en utilisant un certain contrôleur (modèle comportemental) n'est pas définie pendant la phase de conception. Plus tard, dans la phase de mise en œuvre, les batteries intégrées doivent répondre aux exigences du contrôleur conçu. Ce qui limite le choix de la technologie ;

(2) Aucune distinction n'est faite entre des modèles de comportements différents qui fournissent la même qualité de regroupement, car tous les comportements sont considérés comme similaires en termes de consommation énergétique, ce qui n'est pas le cas en réalité [174, 246, 275]. Par exemple, la manipulation consomme plus d'énergie que le déplacement ;

(3) Pour le même modèle de comportements, l'énergie consommée pour accomplir la tâche ciblée peut être très variée en allant d'une expérience à une autre. Parce le choix entre plusieurs actions possibles, consommant différentes quantités d'énergie, et dont l'enchainement aboutisse à un même résultat (formation d'un tas) se fait de manière aléatoire. Par exemple, à chaque étape temps, s'il y a plusieurs cellules vides pour se déplacer ou déposer un objet ou bien prendre un objet parmi plusieurs objets détectés dans l'environnement local, le choix se fait aléatoirement [5, 4].

Par ailleurs, les travaux réalisés pour résoudre le problème d'énergie au sein d'un système de robots en essaim sont déjà discutés dans la section 3.7.2 et présentés sur la figure 4.1. Ils sont dirigés vers le rechargement d'énergie, l'homéostasie énergétique collective ou bien l'optimisation de l'énergie consommée. Dans cette dernière classe, les travaux sont orientés vers la décomposition temporelle de la tâche, la spécification de sous-tâches à des sous-groupes de robots ou bien la limitation d'activités si le robot souffre d'une faible énergie. Le problème d'énergie n'a pas été considéré comme un problème qui incombe au contrôleur de résoudre. Ceci nous a motivé pour tenter de répondre à la question suivante : « Existe-t-il une relation entre le modèle comportemental choisi et l'énergie totale consommée par le système d'un côté et la qualité de formation du tas final d'un autre côté ? »

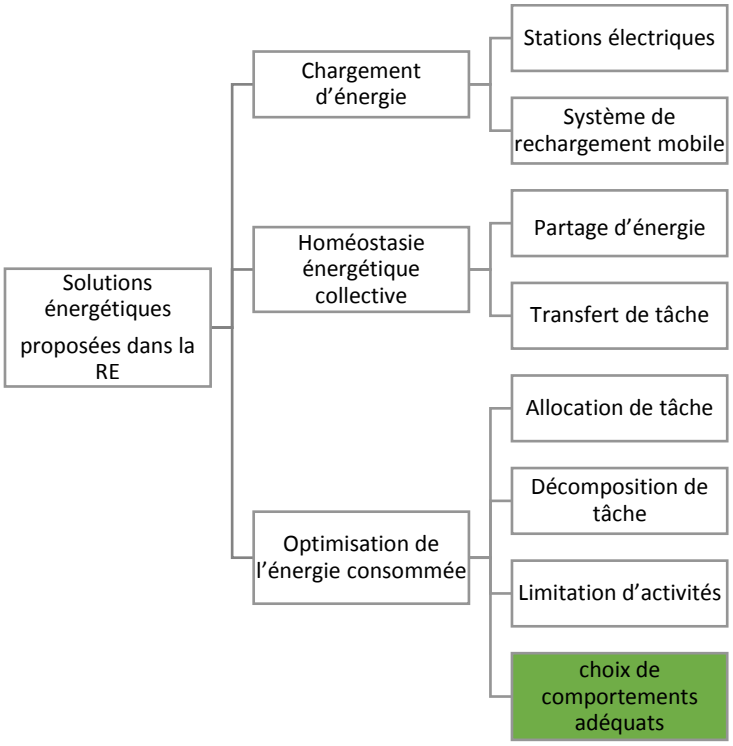


Figure 4.1 : La classification de la solution énergétique proposée

Dans ce travail, on reformule le problème de regroupement d'objets conventionnel, dans lequel les agents-robots regroupent des objets distribués aléatoirement, sans prendre en considération l'énergie consommée pour accomplir cette tâche. La principale contribution de ce travail est la proposition d'une nouvelle méthode d'optimisation d'énergie consommée basée sur le choix de comportement adéquat face à chaque situation dont les robots se trouvent. La proposition décrite dans ce chapitre est classée parmi les solutions déjà proposées afin d'optimiser l'énergie consommée, comme il est présenté sur la figure 4.1.

4.3. Taxonomie du travail proposé

Selon la taxonomie proposée par Brambilla et al. [204], discutée dans la section 3.5, le comportement collectif de regroupement d'objets, étudié dans ce travail, est un comportement d'organisation spatiale. Tandis que la méthode de conception utilisée est l'application d'un algorithme évolutionnaire multi-objectif hors-ligne. Ce qui est catégorisé comme une méthode de conception automatique. Alors que l'analyse ou bien la modélisation est faite en utilisant un modèle microscopique. Où le simulateur NetLogo est utilisé pour décrire la perception, les comportements de base et la méthode de contrôle des agents-robots. La taxonomie de ce travail est présentée sur la figure 4.2.

Et à ce stade, la question qui se pose : pourquoi un simulateur orienté agent et pas un simulateur orienté robot ? En utilisant un simulateur orienté robot, une architecture de robot doit être choisie alors qu'un simulateur orienté agent est plus abstrait en termes d'exigences d'architecture. Le présent travail décrit la relation entre le choix de comportements, la qualité du tas d'un côté et l'énergie consommée de l'autre côté. Et ne s'intéresse pas à définir la quantité d'énergie consommée qui reste relative aux types de robots, d'objets et du terrain utilisés.

Une autre question : pourquoi ce choix de la méthode de conception et de modélisation ? Mermoud et al. [276] ont montré qu'une approche basée sur le modèle macroscopique n'est pas convenable dans tous les cas, et que l'approche basée sur un modèle microscopique semble être plus robuste. Alors que

le choix d'une méthode automatique est dû à la difficulté du problème avec un grand espace de recherche.

Pourquoi un algorithme évolutionnaire et pas une autre méta-heuristique ? L'utilisation d'une méthode d'optimisation dans la robotique en général, est souvent une solution difficile, du fait que la fonction de fitness est, habituellement, bruyante, coûteuse et/ou implicite [50]. Tandis que les autres méta-heuristiques demandent une fonction de fitness bien définie, les algorithmes évolutionnaires traitent les problèmes ayant ces caractéristiques. Dans ce travail, l'algorithme évolutionnaire est appliqué en mode hors-ligne. Comme on l'a déjà mentionné dans la section 3.5, les algorithmes évolutionnaires requièrent un coût de calcul élevé, ce qui n'est pas assuré par un essaim de robots de simple constitution.

Alors qu'il est prouvé que l'utilisation d'une méthode d'optimisation multi-objectif pour résoudre un problème véritablement multi-objectif est mieux que la transformation de ce dernier en un problème mono-objectif [277] en utilisant les méthodes déjà discutées dans le chapitre 1. A cause des bénéfices de l'OMO, plusieurs travaux sont orientés vers «la muli-objectivisation » des problèmes mono-objectif difficiles en le décomposant en plusieurs objectifs [277]. Trianni et López-Ibáñez [277] ont prouvé que l'utilisation d'une méthode d'optimisation multi-objectif est la meilleure option s'il n'y a pas assez d'informations sur le problème à traiter. Ils ont démontré aussi que l'optimisation multi-objectif permet une évolution de comportements plus variés et évite la convergence vers des optimums locaux [50].

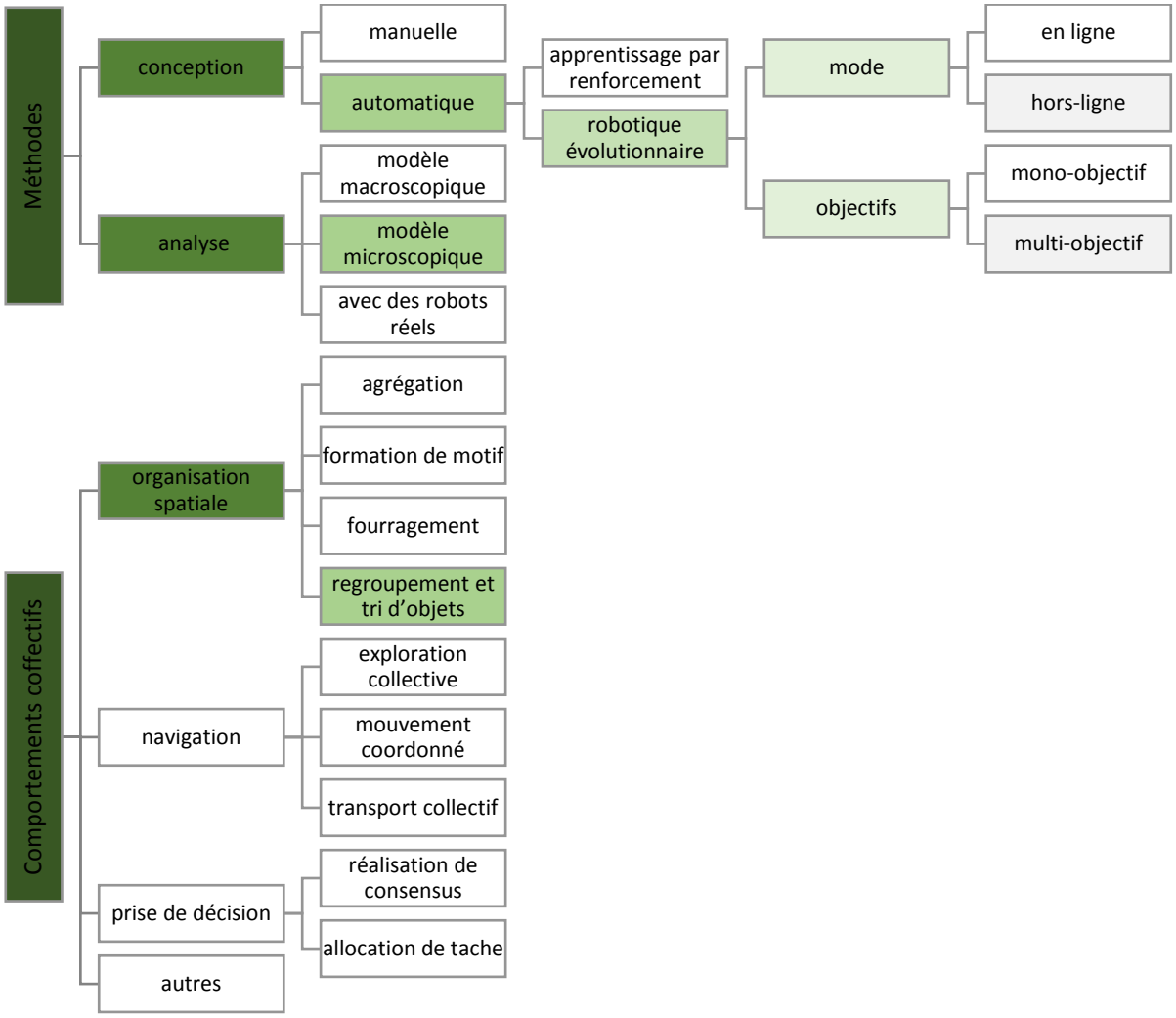


Figure 4.2 : Taxonomie du travail proposé

4.4. Description du problème au niveau macro

Comme on peut le voir sur la figure 4.3, les robots commencent à trouver les objets, les regrouper dans des petits clusters temporels. Par la suite, ces petits clusters vont être fusionnés dans des clusters de plus en plus grands, jusqu'à l'arrivée à un consensus collectif en ce qui concerne la sélection du site de fusionnement de tous les sous-clusters [193].

Le défi lors de la conception d'un système de robots en essaim, est que la tâche à accomplir doit être définie au niveau macro alors que le contrôleur (l'ensemble de règles) est défini au niveau micro [177]. En outre, la compréhension de la façon dont la somme de comportements individuels simples au niveau micro aboutit à un tel comportement global extraordinaire au niveau macro sans que les individus le comprennent, présente un autre défi plus profond [177]. Quelques caractéristiques sont observées au niveau de l'essaim seulement et d'autres sont cachées au niveau de l'individu.

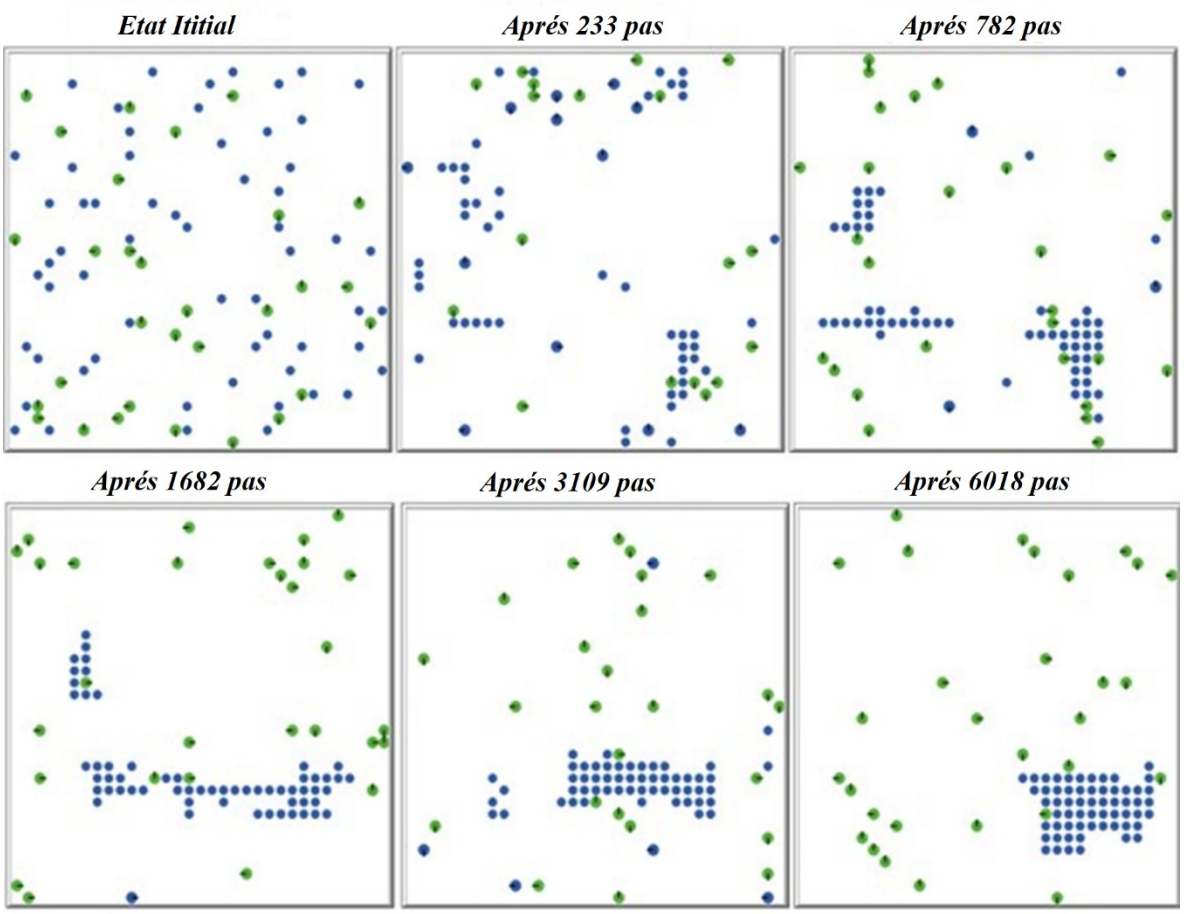


Figure 4.3 : Snapshots de regroupement d'objets (30 robots, 60 objets, dans un environnement 30x30 cellules) [9]

L'objectif de ce travail est de trouver, **au niveau micro**, l'ensemble des règles, ayant la forme : [*si* (perception, état) *alors* comportement], satisfaisant les deux objectifs suivants, définis **au niveau macro** :

4.4.1. La qualité du tas formé

Pour mesurer la qualité du tas formé F_q d'une solution (contrôleur), la moyenne décrite dans l'équation (4.1) est utilisée. Chaque solution est testée I fois, afin de l'évaluer proprement. À chaque entrainement i les conditions initiales sont changées, telles que la position initiale des agents-robots et d'objets ; qui sont distribués aléatoirement à chaque entrainement.

Et pour chaque entrainement i, la fonction de fitness f_{qi}, décrite dans l'équation (4.2), est calculée.

$$Maximiser\ F_q = \frac{\sum_{i=1}^{I} f_{qi}}{I} \tag{4.1}$$

$$f_{qi} = \frac{max\ \{n(C_j)\},\ j=\{1,2,3\ldots J\}}{n_O} \tag{4.2}$$

f_{qi} est calculée comme la proportion de la taille du plus grand cluster au nombre total d'objets. Tel que J est le nombre de clusters construits à la fin de l'entrainement ; $n(C_j)$ est le nombre d'objets dans chaque cluster C_j ; n_O est le nombre total d'objets.

Pour trouver le cluster C_j qui contient le maximum d'objets $max\, n(C_j)$, le maximum d'objets adjacents doit être trouvé. La procédure suivante, décrite dans l'algorithme 4.1, est exécutée pour trouver ce $max\, n(C_j)$ et calculer la qualité du tas formé :

Algorithme 4.1 : Calculer la qualité du tas formé f_{qi}

Les entrées : Les identificateurs de tous les objets sur le terrain ;
(Généralement globaux : accessible dans toutes les procédures)
Les sorties : La qualité du tas $\boldsymbol{f_{qi}}$;

```
01  Construire une liste L1 qui contient les IDs de tous les objets
02  Construire une liste F qui va contenir la taille de tous les clusters
03  Tant que L1 n'est pas vide   // tous les objets sont assignés à un cluster ?
04   Copier l'ID du premier élément de la liste L1 dans une liste vide L2 et le supprimer de la liste L1
05   Pour chaque ID de L2
06      Pour chaque objet dans le voisinage de 4 cellules de cet objet ayant cet ID
07        Si l'ID de cet objet de voisinage n'est pas encore dans la liste L2
08           Copier son ID dans la liste L2 et le supprimer de la liste L1
09        Fin si
10      Fin pour
11  Fin pour
12  Ajouter taille de L2 dans F
13  Fin tant que
14  f_qi = max F / n_O
```

4.4.2. L'énergie consommée

De la même manière, l'énergie consommée F_e est calculée en utilisant la moyenne décrite dans l'équation (4.3). Pour chaque entrainement i, la fonction de fitness de l'énergie f_{ei} est calculée en utilisant l'équation (4.4). f_{ei} est la proportion de l'énergie consommée par tous les robots sur l'énergie initiale totale de ces robots. Tel que n_r est le nombre de robots, e_t est l'énergie initiale donnée à chaque robot au début de chaque entrainement et e_r est l'énergie consommée par le robot r durant l'entrainement.

$$Minimiser\ F_e = \frac{\sum_{i=1}^{I} f_{ei}}{I} \tag{4.3}$$

$$f_{ei} = \frac{\sum_{r=1}^{n_r} e_r}{(n_r\ .\ e_t)} \tag{4.4}$$

L'énergie dans ce travail est considérée comme des unités discrètes comme dans [174, 246, 247].

4.5. La conception du modèle au niveau micro

La conception d'un système de robots en essaim n'est pas une tâche simple. Le processus est généralement divisé en deux étapes : (1) la conception du matériel et (2) la conception du logiciel. Notons que la conception et l'implémentation d'un matériel standard n'est pas un problème, sauf si on a besoin d'une conception très particulière. Alors que la deuxième est généralement une tâche compliquée [50].

Selon Tan et Zheng [192], un modèle général dans la robotique en essaim est divisé en 3 modules : l'échange d'informations, les comportements de base et les comportements complexes. Le module d'échange d'information décrit la façon dont l'information se propage entre les individus de l'essaim, par le biais d'une communication directe (explicite), à travers l'environnement en laissant des traces ou bien par perception. Alors que le module de comportements de base contient les actions de base telles que le déplacement, la manipulation, dépôt, etc. Le module de comportements complexes est nécessaire dans les scénarios beaucoup plus compliqués, afin de simplifier la conception du modèle.

Inspiré par la décomposition proposée dans [192], le modèle utilisé est décomposé en 3 modules, comme il est présenté dans la figure 4.4. Cette division est utilisée dans le but de structurer la section de conception du modèle. Les modules qui existent dans le modèle utilisé dans ce travail sont surlignés. Le modèle se compose d'un module d'échange d'information par perception seulement, un module de comportements de base et un module de contrôle qui représente le module central. Par contre, ce modèle ne contient pas un module de comportements complexes pour la raison que l'essaim est conçu pour faire une seule tâche (le regroupement d'objets).

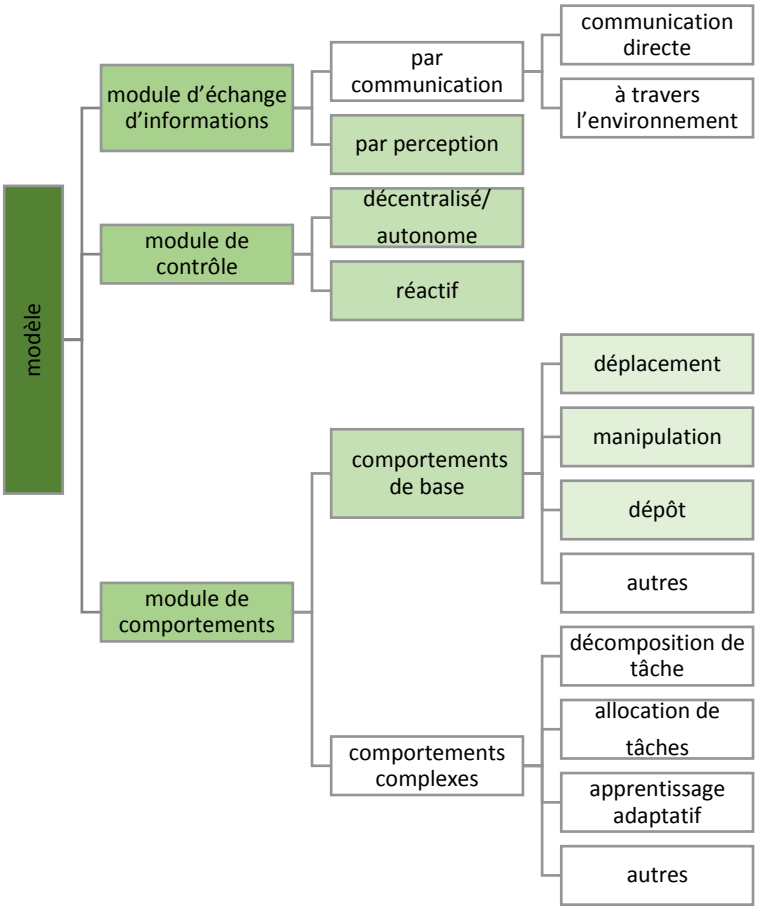
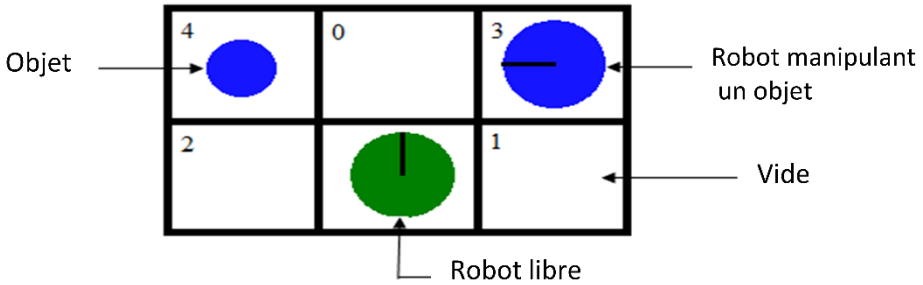


Figure 4.4 : Les modules du modèle utilisé

4.5.1. Module de perception

En premier lieu il faut répondre à la question : pourquoi l'échange d'informations dans le système conçu est basé sur la perception seulement ? L'échange d'informations par communication directe ou bien à travers l'environnement demande : un matériel plus complexe, plus d'énergie et plus de temps [192]. Par contre, la perception ne demande que l'existence de capteurs qui sont des composants essentiels dans l'architecture d'agents-robots. En plus, tant qu'il y a moins de communication, la scalabilité et la robustesse de l'essaim conçu seront plus importantes.

L'environnement dans le simulateur NetLogo est présenté comme une grille de cellules. Les agents-robots ont une vision locale de 5 cellules voisines, comme on peut le voir sur la figure 4.5. La perception de ces cellules et l'état du robot sont codés dans une liste, comme on peut le voir sur la figure 4.5. Chaque cellule peut être vide ou contient un objet ou bien un robot. Tandis que l'état du robot peut être libre ou bien occupé. Le tableau 4.1 résume le codage utilisé pour représenter la perception locale de chaque agent-robot.



La perception du robot libre						Etat
Cellule voisine	0	1	2	3	4	
Valeur	0	0	0	1	2	0

Figure 4.5 : La perception locale d'agent-robot [9]

Dans ce qui suit, on appelle situation la combinaison (perception, état). Les agents-robots peuvent être dans 486 différentes situations, selon la perception possible et l'état d'agent-robot lui-même, variées de [0 0 0 0 0 0] à [2 2 2 2 2 1]. Ce nombre de situations possibles est calculé par :

$$nombre\ d'états\ possible\ d'une\ cellule^{nombre\ de\ cellules} * nombre\ d'états\ possible\ du\ robot = 3^5 * 2 = 486.$$

Tableau 4.1 : Le codage utilisé dans la perception

	État	Codage
Cellule voisine	Vide	0
	Contient un robot	1
	Contient un objet	2
L'agent-robot lui-même	Libre	0
	Occupé	1

4.5.2. Module de comportements de base

Comme les agents-robots ne perçoivent que les 5 cellules voisines, ils ne peuvent opérer que sur ces cellules. On prend l'exemple du robot libre, sur la figure 4.5, il peut se déplacer vers les cellules {0, 1, 2} ou bien manipuler l'objet de la cellule {4}. Dans les travaux précédents, tel que celui de Barfoot et D'eleuterio [4], le choix de la cellule pour se déplacer se fait aléatoirement. Alors que le déplacement

vers les cellules voisines est décomposé en quelques actions fondamentales, comme il est présenté sur le tableau 4.2. Pour les deux cellules {3 et 4}, il existe deux chemins pour se déplacer, à travers la cellule 0 ou bien à travers la cellule {1, 2}, respectivement. Dans ce cas, le robot vérifie d'abord la cellule 0 si elle est libre il prend le chemin à travers cette cellule, ce qui économise l'énergie. On peut déduire que ces déplacements se diffèrent en termes de nombre d'actions fondamentales qu'il faut exécuter pour les accomplir, ce qui veut dire qu'ils ne consomment pas la même quantité d'énergie.

Tableau 4.2 : Décomposition du déplacement en actions fondamentales

Cellule	Actions fondamentales à exécuter
0	1- Avancer.
1	1- Tourner 90° à droite ; 2- Avancer.
2	1- Tourner 90° à gauche ; 2- Avancer.
3	1- Avancer ; 2- Tourner à droite 90° ; 3- Avancer.
	1- Tourner 90° à droite ; 2- Avancer ; 3- Tourner à gauche 90° ; 4- Avancer.
4	1- Avancer ; 2- Tourner à gauche 90° ; 3- Avancer.
	1- Tourner 90° à gauche ; 2- Avancer ; 3- Tourner à droite 90° ; 4- Avancer.

L'énergie nécessaire pour accomplir les actions fondamentales de ce modèle est résumée dans le tableau 4.3. Cette énergie est calculée en gardant la même proportion d'énergie consommée entre les actions fondamentales, utilisée dans ces travaux [174, 246, 275]. Où l'action « ramasser » et l'action « déposer » consomment la plus grande quantité d'énergie. « Avancer » consomme ¾ de l'énergie nécessaire pour « ramasser » ou bien « déposer ». Alors que pour éviter un cas de blocage les agents-robots font un tour de 180°, ce qui est l'équivalant d'un simple évitement dans les travaux déjà cités. Ce simple évitement consomme ½ de l'énergie nécessaire pour « ramasser » ou bien « déposer ».

Tableau 4.3 : L'énergie nécessaire pour chaque action fondamentale

Action fondamentale	L'énergie consommée (unité)
Tourner 90°	1
Tourner 180°	2
Avancer	3
Ramasser	4
Déposer	4

Les comportements de base et l'énergie consommée pour les accomplir sont résumés dans le tableau 4.4. Tel que EC est l'énergie consommée. Elle est calculée en se basant sur les deux tableaux 4.2 et 4.3. Par exemple, pour ramasser l'objet présenté sur la figure 4.5, le robot exécute les actions fondamentales suivantes : 1- Avancer ; 2- Tourner à gauche 90° ; 3- Avancer ; 4- Ramasser. Ce qui veut dire qu'il va consommer $3 + 1 + 3 + 4 = 11\,unités$. L'énergie nécessaire pour tous les autres comportements de base, dans tous les niveaux, est présentée dans le tableau 4.4, est calculée de la même manière.

Tableau 4.4 : L'énergie consommée pour chaque comportement [9]

		Déplacer		**Ramasser**		**Déposer**	
Niveau	Cellule	EC	Codification	EC	Codification	EC	Codification
0	0	3	0	7	3	12	3
1	1 ou 2	4	1	8	4	13	4
2	3 ou 4	7 ou 8	2	11 ou 12	5	20 ou 21	5

Dans ce travail, pour chacun des trois comportements de base, il y a trois niveaux différents, définis selon l'énergie nécessaire pour les accomplir. Chaque niveau d'énergie catégorise les cellules qui sont similaires en fonction du nombre d'actions nécessaires pour opérer sur ces cellules. L'existence de plus d'une cellule et plus d'un chemin dans le même niveau garde une certaine flexibilité. Mais il reste possible que le robot tombe dans une situation de blocage où il ne peut pas exécuter un certain comportement, par exemple, il doit ramasser un objet dans la cellule 3 mais les deux chemins, qui y amènent, sont fermés. Dans ce cas et pour éviter ce blocage, le robot tourne 180° pour changer sa vision, ce qui est similaire à [4].

Il faut noter que les deux comportements de base « ramasser » et « déposer », présentés dans le tableau 4.4, sont différents de ceux présentés dans le tableau 4.3, qui sont des actions fondamentales. La différence c'est qu'un comportement de base « ramasser » veut dire déplacer vers la cellule ensuite exécuter l'action fondamentale « ramasser ». Alors que le comportement de base « déposer », c'est un déplacement vers la cellule, exécution de l'action fondamentale « déposer », ensuite un retour vers la cellule d'origine. Le comportement de base « déplacer » est codifié avec {0,1,2}, correspondant aux niveaux d'énergie qui peuvent être respectivement consommés en exécutant ce comportement. Alors que les deux autres « ramasser » et « déposer » sont codifiés avec {3,4,5}. En codant deux comportements avec le même codage, la question qui se pose est : comment le robot va-il différencier entre les deux ? Ceci dépondéra de son état : si le robot est libre, il va « ramasser » sinon il va « déposer ». Les trois comportements, pouvant être exécutés par chaque agent-robot, sont implémentés sous NetLogo en utilisant les 3 algorithmes : A3.1, A3.2 et A3.3.

4.5.3. Module de contrôle

Le comportement global apparait comme si le système suit un mécanisme très complexe alors qu'il est prouvé qu'on peut simuler ce résultat en intégrant des comportements très simples aux agents-robots [174]. Le défi principal de la conception d'un essaim de robots consiste à concevoir les règles de contrôle individuelles [172], définissant le comportement adéquat à chaque situation. Ces règles peuvent conduire le système à converger vers le comportement global souhaité, qui est plus ce que la somme des comportements de tous les agents-robots. Non seulement le matériel est censé être simple mais aussi le logiciel (contrôleur) [177]. À cet effet, l'approche de contrôle standard, dans la robotique en essaim, est donc basée sur le contrôle réactif [177], en utilisant le modèle le plus simple d'agent réactif. Un agent réactif n'a que des règles simples de type (**Si** situation X **alors** comportement Y) représentées dans la table de correspondance (se référer au tableau 4.5), où les comportements adéquats aux 486 situations, représentés par « ? » sont sujet de recherche.

Tableau 4.5 : Table de correspondance situation/ comportement

Situation S		**Comportement**
Perception S [0-4]	**État S [5]**	**C ∈{0, 1, 2, 3, 4, 5}**
00000	0	?
00000	1	?
00001	0	?
…	…	…
11021	0	?
11021	1	?
11022	0	?
…	…	…
22221	1	?
22222	0	?
22222	1	?

Quoique le système de contrôle centralisé ait l'avantage d'être plus rapide et précis puisqu'il prend en main tous les agents-robots du système [174] ; le système de robots en essaim est doté d'un contrôle distribué [172, 177]. Pour la raison que l'existence d'un chef d'équipe ou d'un plan à suivre se confond avec le principe d'auto-organisation qui caractérise ce type de systèmes [172]. En plus, le contrôle distribué a montré qu'il est plus efficace avec les systèmes peuplés et complexes, ce qu'est le cas dans les systèmes de robots en essaim [172]. Ce qui veut dire que le contrôle décrit dans l'algorithme 4.2 est exécuté par chaque agent-robot dans le terrain, d'une manière séparée et autonome.

Algorithme 4.2 : Le contrôle

Les entrées : La situation actuelle **S : liste ;**
Les sorties : Le comportement à exécuter **C ∈{0, 1, 2, 3, 4, 5} ;**
Appel à Déplacer, Ramasser ou bien Déposer ;

```
01  C = Correspondant (S) ; //chercher dans la table de correspondance le comportement correspond
02  à la situation S //
03  Si C ∈{0, 1, 2} Alors Déplacer (S, C) ; // Algorithme A3.1 //
04  Sinon
05    Si S [5] = 0 Alors Ramasser (S, C) ; // Algorithme A3.2 //
06    Sinon
07      Déposer (S, C) ; // Algorithme A3.3 //
08    Fin si
09  Fin si
```

4.6. La structure d'un agent-robot

Après avoir détaillé les trois modules de perception, comportements de base et contrôle, on peut résumer l'architecture des agents-robots comme il est représenté sur la figure 4.6. L'architecture structurale et fonctionnelle des agents ne doit pas être très compliquée. En effet, des agents réactifs avec une vision limitée et une interaction simple avec leur voisinage, et n'ayant pas des capacités de calcul ou de mémorisation, sont capables d'accomplir la tâche accordée [230].

Les entrées/sorties de notre système de contrôle sont supposées être discrètes, ce qui nécessite des pré/post traitements (voir la figure 4.7). Pour identifier nos agents-robots à des robots réels, on a supposé que des pré-traitements sont déjà faits sur les données brutes des senseurs (capteurs) afin d'obtenir des capteurs virtuels orthogonaux (indépendants) [193, 4] fournissant des valeurs discrètes. En outre, au lieu de préciser la position ou bien la vitesse d'actionneurs, un comportement simple est sélectionné, à partir d'un ensemble fini de comportements, comme sorti du système de contrôle. Par la suite, une étape de post-traitement doit être faite pour convertir le comportement sélectionné en un ensemble de commandes réelles. Après avoir conçu les deux étapes de pré/post-traitements, l'étape suivante est de trouver le comportement adéquat pour chaque situation (perception + état de robot), ceci est détailler dans ce qui suit.

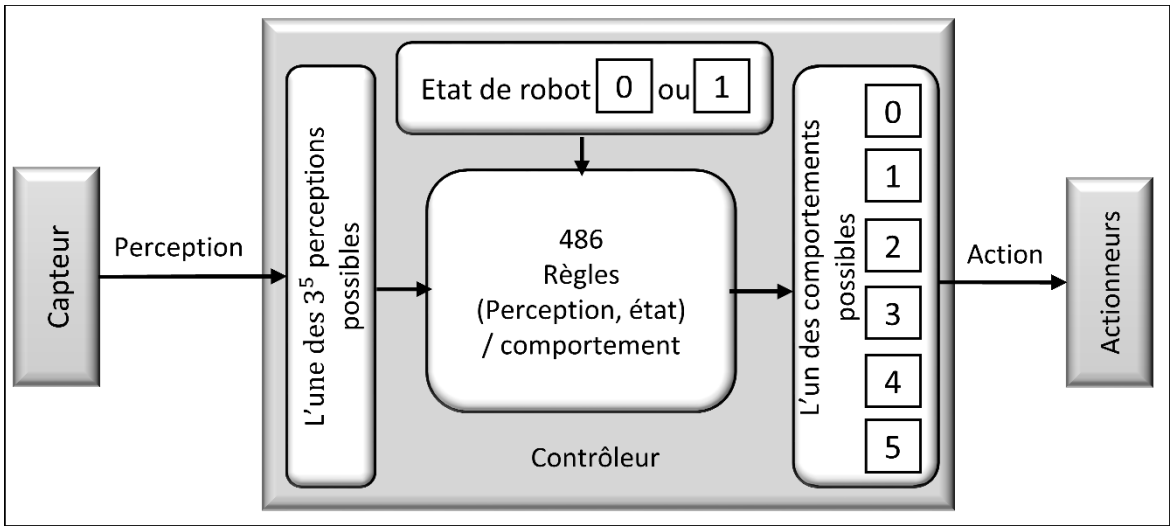


Figure 4.6 : La structure des agents-robots

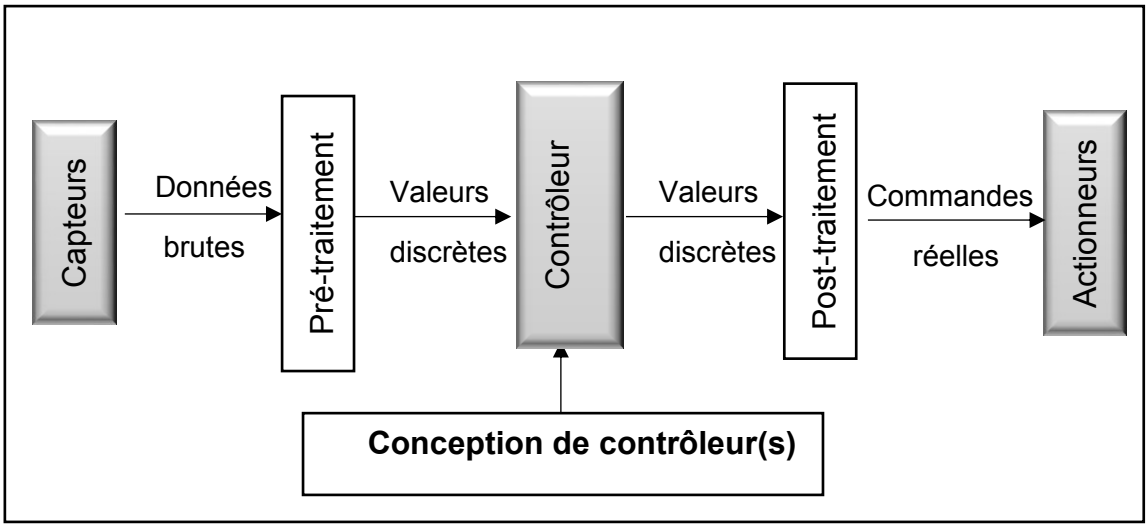


Figure 4.7 : La décomposition en 3 étapes

L'utilisation de cette approche de décomposition de la phase de conception d'agents-robots en 3 étapes nous permet d'avoir des contrôleurs indépendants de l'architecture physique des robots utilisés, puisque les entrées/sorties sont discrètes et indépendantes de l'architecture. Si le contrôleur adéquat est trouvé, il suffit de se concentrer sur les deux étapes de pré et post-traitement pour mettre en œuvre ce contrôleur dans un robot physique choisi. En cas de changement de mode de manipulation d'objet, par exemple, la pince par une pelle, le concepteur refait seulement l'étape de post-traitement, et non pas l'étape de conception de contrôleur, qui prend beaucoup de temps de calcul.

L'objectif de ce travail repose sur la conception de contrôleur(s) adéquat(s), qui nous permet(tent) d'avoir une bonne qualité du tas formé, et en même temps il(s) consomme(nt) moins d'énergie. Un algorithme évolutionnaire peut prendre en charge le rôle de conception automatique de ces contrôleurs [193]. Pour chaque situation dans la table de correspondance (tableau 4.5), il faut trouver le comportement adéquat, pour construire les 486 règles de la forme (**si** situation X **alors** comportement Y). L'ensemble de ces 486 règles représente un contrôleur.

4.7. Description du résolveur

Même si les contrôleurs eux-mêmes sont simples, il est difficile de les trouver [177], et pour cette raison le calcul évolutionnaire est utilisé. Il n'est pas toujours évident dans un système multi-agent de prédéterminer explicitement l'ensemble des règles qui forme le contrôleur souhaité. Par contre l'approche utilisée sert à identifier les objectifs globaux. Ensuite, les règles de contrôle nécessaires pour atteindre ces objectifs sont sujettes de recherche. Dans ce cadre, deux fonctions de fitness sont utilisées pour effectuer des mesures globales de la performance du système, ce qui facilite la sélection des contrôleurs [193]. De cette manière, l'ensemble de règles est défini implicitement.

Au début, une population initiale de N individus (solutions), ayant la structure décrite dans la section 4.7.1, est générée. Où le chromosome de chaque individu $Chrom\,[1-486] = aléatoire\,\{0,1,2,3,4,5\}$. La section 4.7.2 décrit l'étape suivante d'évaluation du comportement collectif, résultant de l'utilisation du chromosome de chaque individu comme contrôleur, en utilisant les fonctions de fitness moyennes F_q et F_e. Un rang est assigné à chaque individu en utilisant la procédure décrite dans la section 4.7.3. Ensuite la distance d'encombrement (section 4.7.4) est calculée entre les individus du même rang. À ce niveau, un ordre total entre les individus peut être établi. Une nouvelle population est générée suivant la procédure décrite dans la section 4.7.5. Ce processus est répété jusqu'à atteindre la condition d'arrêt. Le détail de l'algorithme est donné dans la section 4.7.6.

4.7.1. Structure des individus

Chaque individu (solution) de la population est représenté par la structure décrite ci-dessous.

```
Individu : structure
01  {
02   Chrom : tableau [486]          // contient les 486 comportements (le contrôleur)
03   F_q, F_e : réels               // les deux valeurs de fitness de l'individu
04   Rank : entier                  // classement de l'individu
05   Dominationset : liste          // l'ensemble d'individus qu'il domine
06   Dominatedcount : entier        // le nombre d'individus qui le dominent
07   Crowddist : réel               // la distance d'encombrement
08  }
```

4.7.2. L'évaluation d'un individu

L'utilisation d'une fonction de fitness, permet d'entrainer les contrôleurs [193] représentés par les chromosomes. Puisque les fonctions de fitness F_q et F_e indiquent le rendement d'un certain contrôleur face à la tâche à effectuer, elles permettent aux contrôleurs, qui ont un bon fonctionnement, d'avoir un bon rang lors de la phase de classement (Algorithme A4.1). Comme il est décrit dans la procédure d'évaluation (Algorithme 4.3), des valeurs moyennes de fitness sont déterminées en répétant l'expérience d'entrainement avec le même contrôleur plusieurs fois, afin de donner une mesure plus ou moins stable (avec le minimum de marge d'erreur). A chaque entrainement, les conditions initiales de l'environnement sont générées aléatoirement. Dans ce travail, ces conditions sont les positions et les orientations des agents-robots et des objets. Comme il est décrit dans l'algorithme 4.4, les agents-robots utilisent le chromosome de l'individu comme contrôleur, pendant un certain temps T fixé pour tous les entrainements et tous les individus.

4.7.3. Le classement

Suivant le principe de dominance, décrit dans le chapitre 1, les individus sont classés dans des fronts non-dominés selon leurs rangs, comme il est présenté dans l'algorithme A4.1. Pour assigner ce rang à un individu, le nombre d'individus qui le dominent est calculé, ainsi que les individus qu'il domine sont trouvés. Le rang 1 est assigné aux individus qui ne sont pas dominés du tout, le rang 2 est assigné aux

individus dominés, seulement, par ceux du rang 1, et les individus du rang 3 sont dominés par ceux du rang 1 et 2. Le processus décrit dans l'algorithme A4.1, linge 20 à 26, est répété jusqu'à ce que tous les individus sont classés dans des fronts non-dominés.

Algorithme 4.3 : L'évaluation

Les entrées : Un individu : structure
Les sorties : Un individu : structure avec changements dans $\boldsymbol{F_q}$, $\boldsymbol{F_e}$;

```
01  F_q = 0.0 ; F_e = 0.0 ; Deads = 0 ;
02  Pour Nombre d'entrainements I          // conditions initiales
03    L'entrainement (Chrom) ;             // Algorithme 4.4
04    Calculer f_qi ;                      // Algorithme 4.1
05    Calculer f_ei ;
06  Fin pour
07  Calculer la moyenne F_q, F_e           // la moyenne de I entrainements
```

Algorithme 4.4 : L'entrainement

Les entrées : Chrom : tableau // le chromosome de l'individu ;
Les sorties : Changement dans l'environnement ;

```
01  Réinitialisation aléatoire de l'environnement ;
02  Pour temps d'entrainement T
03    Pour chaque robot
04     Perception ;
05     Contrôle (S) ;        // Algorithme 4.2 en utilisant Chrom comme contrôleur
06    Fin pour
07  Fin pour
```

4.7.4. La distance d'encombrement

Après avoir classé les individus dans des fronts non-dominés, il faut différencier entre les individus du même front (ayant le même rang). À cet effet, une distance d'encombrement est calculée pour maintenir la diversité. La procédure de calcul de la distance d'encombrement est décrite dans l'algorithme A4.2. Pour chaque fonction objectif k, dans notre cas $k = \{q\ ou\ e\}$, les individus du même front sont triés par ordre croissant selon la fonction objectif k. Les deux individus $1\ et\ l$ à l'extrémité de la liste d'individus triée L ont un $Crowddist_k^1 = 1, Crowddist_k^l = 1$. Pour chaque individu i parmi les individus classés entre $2\ et\ l-1$:

$$Crowddist_k^i = \frac{F_k^{i+1} - F_k^{i-1}}{F_k^l - F_k^1} \qquad (4.5)$$

La distance d'encombrement d'un individu i est calculée par la somme des distances d'encombrement (équation 4.6) calculées pour chaque objectif par l'équation (4.5).

$$Crowddist^i = \sum_{k=q}^{k=e} Crowddist_k^i \qquad (4.6)$$

Il faut noter que la valeur associée aux deux individus d'extrémité dans l'algorithme original est ∞. Cette valeur est utilisée pour que l'algorithme soit flexible et utilisable avec tous les problèmes. Dans notre cas, le simulateur NetLogo n'est pas trop dédié aux calculs mathématiques. On a utilisé la valeur 1, qui est la plus grande distance qu'on peut avoir entre deux valeurs d'une fonction objectif.

4.7.5. Génération d'une nouvelle population

Pour générer la nouvelle population, certains individus doivent être sélectionnés pour y participer. La sélection dans l'algorithme original de NSGA-II est une sélection par tournoi binaire basé sur le principe suivant : sélectionner 2 individus aléatoirement, l'individu participant sera celui qui a le

meilleur rang entre les deux, s'ils ont le même rang c'est celui avec la plus grande distance d'encombrement. A cet effet et quand la probabilité de l'opérateur génétique (croisement, mutation) est importante, un individu peut être sélectionné plusieurs fois. Tandis que l'intensification est garantie par la sélection de N meilleurs individus, on a besoin de plus de diversification, puisque l'espace de recherche est large. Pour cette raison, un ensemble d'individus non répétés est sélectionné pour chaque opérateur, comme il est présenté dans l'algorithme A4.3, linge 2 et linge 11. La manière dont les deux opérateurs génétiques, croisement et mutation, sont appliqués sur les chromosomes des individus sélectionnés est décrite dans les algorithmes A4.4 et A4.5, respectivement. Avec un ensemble de valeurs discrètes, on a utilisé une mutation cyclique.

4.7.6. L'algorithme principal

NSGA-II implémente l'élitisme sans avoir besoin d'une archive externe [278]. L'ancienne population et la nouvelle sont mélangées (ligne 18, Algorithme 4.5), pour sélectionner les N meilleurs individus (linge 23, Algorithme 4.5). Cette sélection est devenue possible après avoir établi un ordre total entre les individus. Les individus sont triés selon leur rang par ordre croissant et selon la distance d'encombrement par ordre décroissant, afin de différencier les individus ayant le même rang. Dans l'algorithme NSGA-II conventionnel, après avoir atteint le maximum d'évaluations, l'algorithme se termine et les individus non-dominés sont rendus au décideur pour y choisir celui qui lui convient.

Algorithme 4.5 : L'algorithme principal du solveur

Les entrées : N : entier ; pCrois, pMut, mu : réels ;
Les sorties : Front de Pareto ;

```
01  Pop = liste des ID de N individus générés aléatoirement ;          // population initiale
02  Pour i = 1 à N
03    [Individu Pop (i)] = Evaluation (individu Pop (i)) ;             //Algorithme 4.3
04  Fin pour
05  FE = N ;                                                          // nombre d'évaluations
06  [Pop, F] = Non-dominated Sorting (Pop) ;                          //Algorithme A4.1
07  [Pop] = Crowding distance (Pop) ;                                 //Algorithme A4.2
08  [Pop] = Tri décroissant des individus selon la distance d'encombrement (Pop) ;
09  [Pop] = Tri croissant des individus selon le rang (Pop) ;
10  Tant que FE < Max-Eval
11    nCrois = 2 * entier ( pCrois * N / 2) ;
12    nMut = entier ( pMut * N) ;
13   [New-pop] = Génération d'une nouvelle population (Pop, nCrois, nMut, mu) ;//Algorithme A4.3
14    Pour i = 1 à taille (New-pop)
15      [Individu New-pop(i)] = Evaluation (individu New-pop(i));
16    Fin pour
17    FE = FE + taille (New-pop) ;
18    Pop = Pop + New-pop ;                    // mélanger l'ancienne avec la nouvelle population
19    [Pop, F] = Non-dominated Sorting (Pop);
20    [Pop] = Crowding distance (Pop) ;
21    [Pop] = Tri décroissant des individus selon la distance d'encombrement (Pop) ;
22    [Pop] = Tri croissant des individus selon le rang (Pop) ;
23    [Pop] = Tronquer les N premiers (Pop) ;          // garder les N meilleurs individus seulement
24     // mettre à jour
25    [Pop, F] = Non-dominated Sorting (Pop) ;                        //Algorithme A4.1
26    [Pop] = Crowding distance (Pop) ;                               //Algorithme A4.2
27    [Pop] = Tri décroissant des individus selon la distance d'encombrement (Pop) ;
28    [Pop] = Tri croissant des individus selon le rang (Pop) ;
29  Fin tant que
    Représenter les individus ayant Rank = 1 ;
```

4.8. Résultats de NSGA-II

L'algorithme décrit ci-dessus est appliqué sur le problème de regroupement d'objets bi-objectif avec 60 objets, 30 agents-robots dans un terrain d'entrainement de 30x30 cellules, comme le montre la figure 4.8. L'algorithme NSGA-II est exécuté 10 fois avec les paramètres suivants, fixés empiriquement : temps d'entrainement 2000 pas de temps, les agents-robots ont une énergie initiale égale à 9000 unités, le nombre de conditions initiales I est 10, le nombre maximal d'évaluations Max-Eval est égal à 30000, la taille de la population N est égale à 65, le pourcentage de croisement est 0.9, le pourcentage de mutation est 0.4 et le taux de mutation est 0.03.

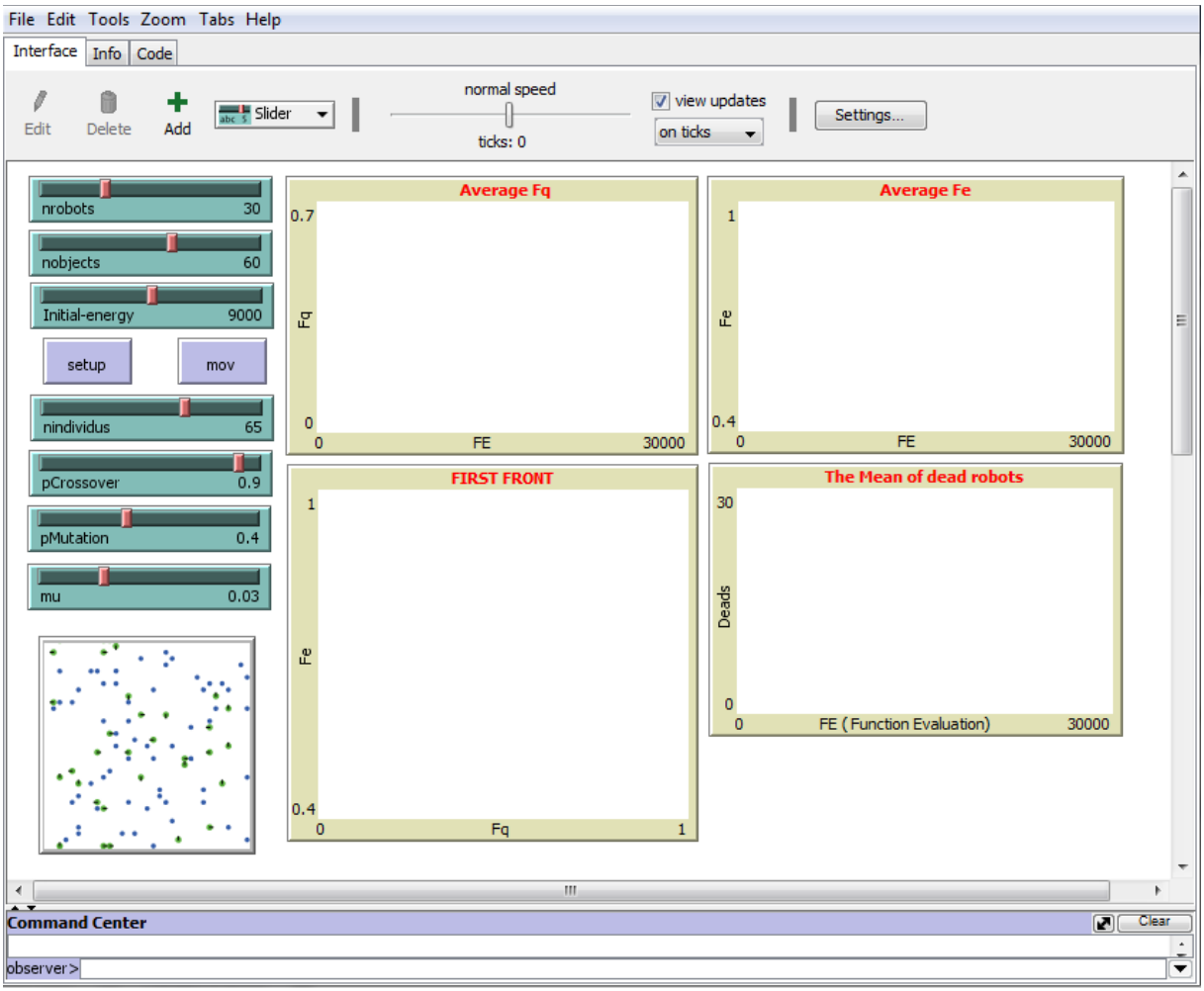


Figure 4.8 : L'environnement de simulation NetLogo

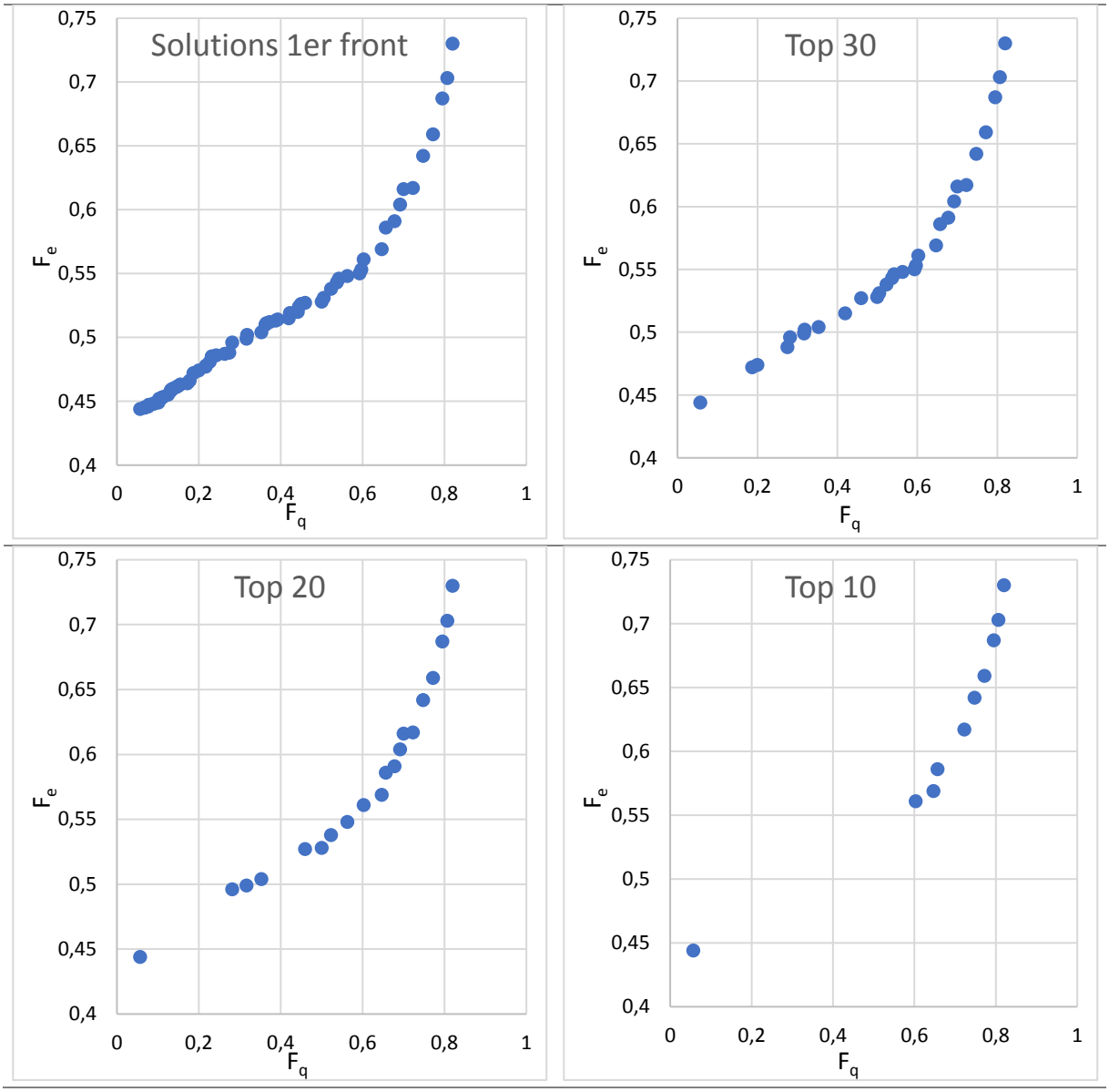


Figure 4.9 : L'approximation du front de Pareto obtenue par NSGA-II

La figure 4.9 présente les meilleures solutions obtenues en utilisant NSGA-II durant les 10 exécutions. Ces solutions (modèles comportementaux) dominent les autres solutions mais elles ne se dominent pas entre elles. Ce qui prouve que le problème présenté est un problème multi-objectif avec deux objectifs contradictoires. Certains modèles comportementaux sont meilleurs que les autres en termes de qualité de formation des tas et d'autres sont meilleurs que les autres en termes d'économie d'énergie. Et, comme il n'existe pas de modèle comportemental qui satisfait les deux objectifs en même temps, ces solutions constituent le meilleur compromis possible trouvé entre les deux objectifs. Ces compromis sont fournis au décideur pour choisir le modèle comportemental qui répond à ses besoins. Pour faciliter la tâche, les 30, 20 et 10 meilleures solutions sont aussi présentées sur la figure 4.9.

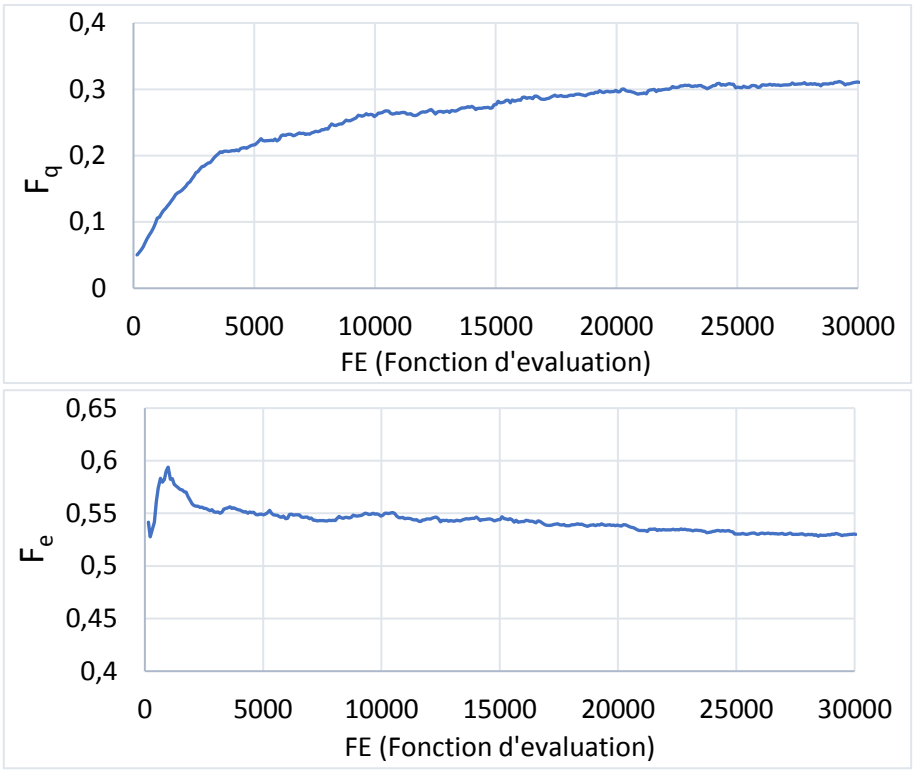


Figure 4.10 : La vitesse moyenne de convergence des deux objectifs durant 10 exécutions en utilisant NSGA-II

L'évolution moyenne de F_q et F_e de la population durant la recherche évolutive, est représentée sur la figure 4.10. Au début, $FE \leq 317$, et avec une population initiale aléatoire, les agents-robots effectuent certaines actions qui consomment de l'énergie mais ne donnent pas une amélioration significative en termes de qualité de regroupement ; la majorité des chromosomes ne correspondent pas à la perception de l'environnement local. Ainsi, la plupart des agents-robots tournent autour d'eux à 180° tout en consommant 2 unités d'énergie à chaque pas de temps, donc aucun tas n'est formé. Après cela (317< FE≤ 16900), NSGA-II réussit à améliorer la qualité du tas formé, sachant que la moyenne de l'énergie consommée diminue. Jusqu'à ce point, l'algorithme utilisé se concentre sur la stratégie de diversification pour explorer de plus grandes zones de l'espace des solutions. Dans les étapes plus avancées (FE > 16900), cet algorithme limite son espace de recherche à une petite zone, adoptant ainsi une stratégie d'intensification, lui permettant d'exploiter les chromosomes trouvés. Dans cette dernière étape, F_q augmente plus lentement et F_e est plus stable et diminue aussi légèrement que possible ; cet algorithme tend alors à converger.

4.9. Conclusion

Ce chapitre présente un problème d'optimisation bi-objectif concernant la tâche de regroupement d'objets. L'analyse de la littérature montre qu'avec une tâche aussi compliquée, l'énergie devient la principale limitation pour accomplir le travail sans perdre un certain nombre de robots, en raison de leurs batteries déchargées, ce qui conduit à un possible avortement de la mission. Pour cette raison, le problème habituel du regroupement d'objets à un seul objectif est remodelé pour traiter deux objectifs conflictuels : regrouper les objets distribués de la manière le plus large possible et conserver au mieux l'énergie des robots. L'algorithme évolutif multi-objectif NSGA-II est appliqué. Les résultats montrent que les bonnes solutions en termes de qualité de regroupement consomment plus d'énergie, mais qu'elles restent le meilleur compromis possible dans ce contexte.

Chapitre 5: Étude de l'un des meilleurs modèles comportementaux bi-objectif trouvés

5.1. Introduction

Toutes les solutions non-dominées présentées dans le chapitre précédent sont les meilleurs compromis trouvés en termes de qualité de regroupement et de consommation d'énergie. Dans un premier temps, l'une des meilleures solutions représentant un modèle comportemental bi-objectif (MC-BO) est sélectionnée pour être évaluée dans différentes circonstances et examiner la robustesse, la scalabilité et la flexibilité du système en utilisant ce MC-BO. Quelques facteurs dynamiques et accidentels peuvent influer sur le fonctionnement du système, pour cette raison, leur effet doit être étudié. Pour prouver la supériorité de l'approche proposée, deux modèles comportementaux bi-objectif sont comparés à un autre mono-objectif.

Le reste de ce chapitre est organisé comme suit. La section 5.2 analyse le modèle comportemental bi-objectif MC-BO. L'effet de la capacité de batterie est étudié dans la section 5.3. Dans la section 5.4, la robustesse du système est examinée. Et dans la section 5.5, sa scalabilité est sujette d'investigation. Ensuite, pour examiner sa flexibilité, l'effet du nombre d'objets, de la taille de l'environnement et la taille du problème sont étudiés dans la section 5.6. Alors que dans les sections 5.7 et 5.8, quelques facteurs dynamiques et accidentels, qui peuvent avoir une influence sur le fonctionnement du système, sont discutés. Dans la section 5.9, quelques scénarios sont imaginés pour examiner l'effet de ces facteurs. Tandis que la section 5.10 présente une comparaison entre deux modèles comportementaux bi-objectif et un autre mono-objectif afin de prouver la supériorité de l'approche proposée.

5.2. Analyse du modèle comportemental bi-objectif ciblé (MC-BO)

Parmi les 486 comportements représentant le modèle comportemental, 12 comportements différents illustrés dans le tableau 5.1 sont choisis pour être expliqués plus en détail. Dans les situations {1}, {2}, {3} et {4}, il y a plus d'une cellule vide ; le meilleur comportement de déplacement possible qui consomme moins d'énergie est sélectionné. Dans la situation {3}, par exemple, l'agent-robot a 3 cellules voisines vides : les numéros 1, 3 et 4. Parmi les comportements de déplacement possibles 1 et 2 consommant respectivement 4 et 8 unités, 1 est sélectionné. De plus, dans les situations {7} et {9}, le robot dispose de plus d'un objet. Le comportement de ramassage exécuté permet de conserver, respectivement, 1 et 4 unités, dans chaque situation. Dans les situations {8} et {10}, le comportement de dépôt sélectionné permet de conserver 8 unités dans les deux situations. En revanche, dans les situations {5}, {6}, {11} et {12}, les agents-robots exécutent les comportements 2 et 5. Même si ces comportements consomment plus d'énergie que les autres, personne ne peut prouver qu'ils n'ont aucun effet sur l'émergence finale du tas. Dans la situation {11}, l'agent-robot ne perçoit qu'un seul objet. La probabilité que cet objet appartienne à un sous-groupe est faible, donc le ramassage de cet objet peut améliorer la qualité. Alors que dans la situation {12}, l'agent-robot dispose de 2 cellules vides pour déposer son objet. Le comportement sélectionné est celui qui consomme le plus d'énergie. Mais, en déposant l'objet dans la cellule 4, une collection de 3 objets adjacents sera construite. Cela peut étendre un groupe possible que l'agent-robot ne peut pas percevoir ou construire en un sous-groupe de 3 objets dans le pire des cas. Bien que MC-BO consomme plus d'énergie que les autres solutions présentées dans la figure 4.9, l'analyse de ses comportements montre qu'il parvient à économiser de l'énergie tout en restant le meilleur d'un point de vue qualité de formation.

Tableau 5.1 : Comportements représentatifs de MC-BO

Situation	Perception	État	Comportement sélectionné
1	**0**0000	0	0
2	**0**0012	1	0
3	2**0**200	0	1
4	22**0**01	1	1
5	022**0**2	0	2
6	0221**0**	1	2
7	**1**1202	0	3
8	**0**1101	1	3
9	00**1**10	0	4
10	21**0**10	1	4
11	2002**1**	0	5
12	1201**0**	1	5

L'analyse des comportements représentatifs de MC-BO a montrée qu'il existe des solutions non dominées qui offrent une bonne qualité de regroupement tout en consommant plus d'énergie que les autres solutions non dominées, mais elles restent meilleures que les solutions dominées. Cette analyse ira encore plus loin, en évaluant MC-BO dans différentes conditions limites dans les sections suivantes.

5.3. L'effet de la capacité de batterie

Pour être dans les conditions expérimentales réelles, l'énergie des agents-robots ne peut être considérée comme illimitée ou suffisante pour accomplir la tâche. Ainsi, MC-BO (la solution à tester), est mise sous des conditions limites d'énergie pour prouver que même si ces conditions ont un effet sur les performances des agents-robots, le système continu à fonctionner encore pendant une durée

suffisante, lui permettant d'accomplir la tâche. Cela est possible parce que MC-BO est supposé permettre au système robotique d'économiser l'énergie autant que possible.

Différentes capacités de batterie (CB=18000, 15000, 12000 unités) sont testées. La figure 5.1 montre la moyenne de F_q sur 500 simulations avec des conditions initiales aléatoires et 5000 pas de temps dans chaque simulation. Le nombre d'agents-robots morts est également présenté. Les agents-robots morts ont une énergie nulle dans leurs batteries [279]. Les agents-robots ayant un CB=18000 unités comme énergie initiale peuvent accomplir les 5000 pas sans aucun agent mort et F_q atteint 0,8, tandis que pour CB=15000 unités, les agents-robots commencent à mourir après 4482 pas de temps. Pour CB=12000 unités, les agents-robots commencent à mourir plus tôt, après 3523 pas de temps. La figure 5.1 montre que plus la capacité de la batterie des agents-robots diminue, plus les performances du système se dégradent.

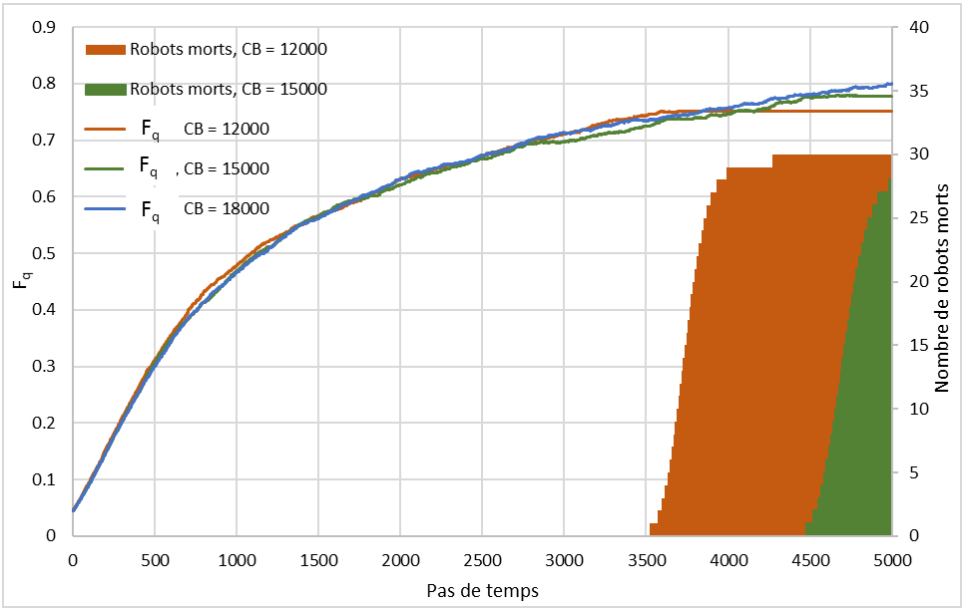


Figure 5.1 : L'effet de la capacité de batterie sur les performances du système et le nombre d'agents-robots morts

5.4. La robustesse

Pour étudier la robustesse du système et trouver le seuil des agents-robots morts pour lequel le système n'est plus opérationnel, nous devons nous concentrer sur le moment où les agents-robots commencent à tomber en panne. La figure 5.2 montre que F_q converge toujours et que le système reste robuste (dans les deux cas : CB = 15000 et CB = 12000 unités) tant que moins de 86% des agents-robots sont morts (26 agents-robots morts sur un total de 30). Cette grande robustesse est due (1) à la simple constitution des agents-robots et à leur ressemblance, impliquant l'homogénéité de leur système, qui conduit à leur abondance, de sorte que les agents-robots restants peuvent couvrir les agents-robots morts, bien que les performances du système avant et après l'apparition du phénomène de mort soient différentes (2) le système est totalement décentralisé, ce qui permet de continuer à fonctionner sans avoir besoin de se coordonner.

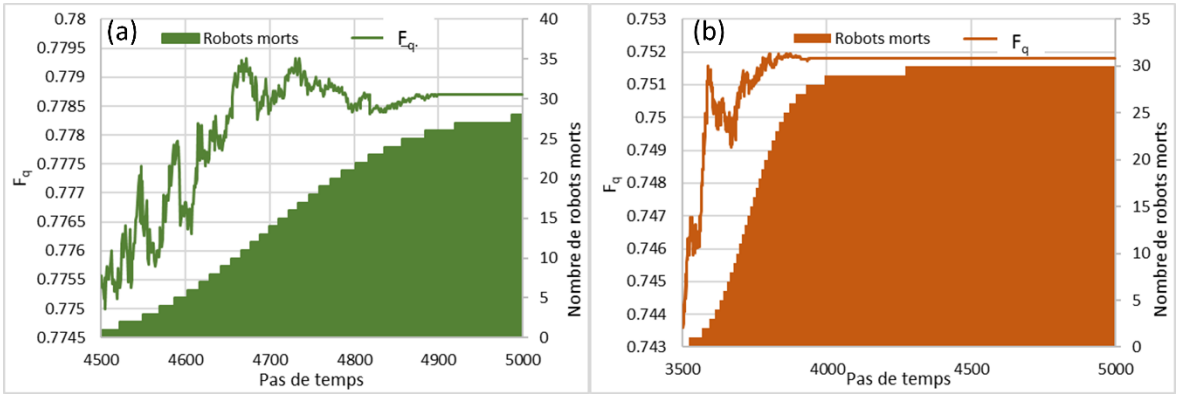


Figure 5.2 : La robustesse du système avec (a) CB = 15000 unités (b) CB = 12000 unités

5.5. La scalabilité

Pour examiner la scalabilité du système, le nombre d'agents-robots est varié, tandis que le nombre d'objets et les dimensions du terrain d'entraînement sont maintenus constants (nombre d'objets = 60, dimensions du terrain d'entraînement = 33 x 33), sachant que les agents-robots ont une capacité de batterie de 18 000 unités. Dans chaque expérience, le modèle comportemental ciblé est testé 500 fois (avec des conditions initiales aléatoires à chaque test), sachant que le critère d'arrêt est $F_q \geq 0{,}9$ ou $F_e \geq 0{,}98$. La figure 5.3 montre que dans une gamme de 21 à 60 agents-robots, les performances du système sont plus stables, avec un minimum d'agents-robots morts, ce qui signifie que ce système est scalable dans cette gamme, tandis qu'en dehors de cette gamme, il ne peut pas bien fonctionner. Les agents-robots continuent à fonctionner jusqu'à ce qu'ils consomment 98 %, en moyenne, de leur énergie, sans réussir leur mission. Pour bien faire avec peu d'agents-robots, ils doivent être équipés de batteries de très grande capacité. De plus, les performances se dégradent avec un terrain d'entraînement surpeuplé, en raison d'une forte quantité d'interférences entre les agents-robots.

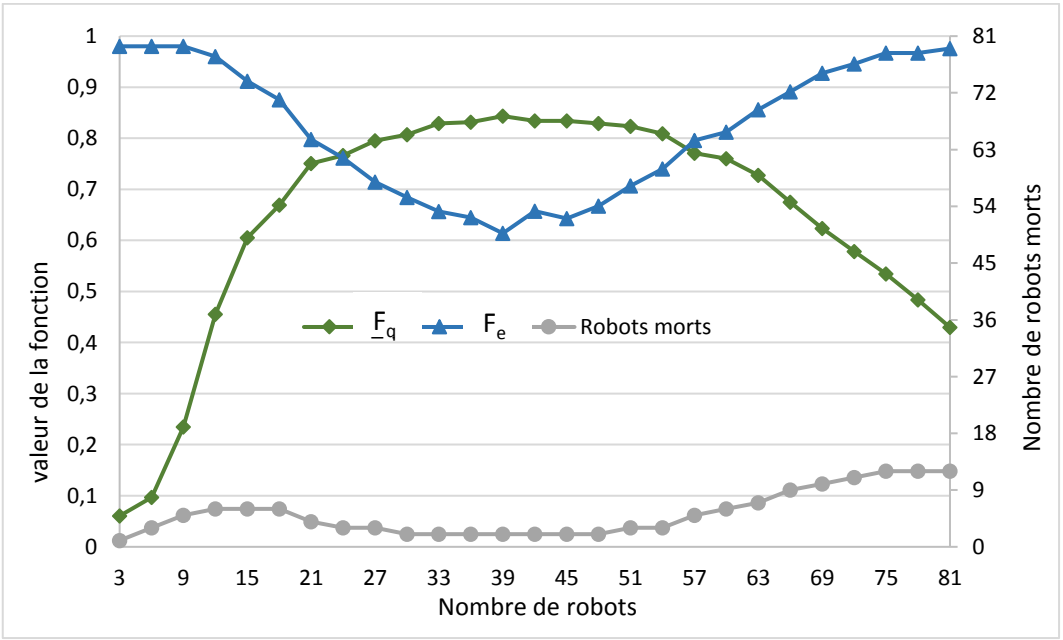


Figure 5.3 : La scalabilité du système utilisant MC-BO

5.6. La flexibilité

Les trois expériences suivantes sont réalisées pour examiner la flexibilité du système dans des conditions de consommation d'énergie. Les résultats présentés sont les moyennes de 500 simulations avec une énergie initiale =18000 unités et le critère d'arrêt pour chacune est $F_q \geq 0.9$ ou $F_e \geq 0.98$.

5.6.1. L'effet du nombre d'objets

Pour examiner l'effet du nombre d'objets sur les performances du système, seul ce nombre est modifié alors que le nombre d'agents-robots et les dimensions du terrain d'entraînement sont fixes (nombre d'agents-robots=30, dimensions du terrain d'entraînement=33x33). La figure 5.4 montre que pour un nombre d'objets inférieur ou égal au nombre d'agents-robots, tous les objets sont ramassés et le processus de collecte ne peut pas démarrer et aucun tas ne peut être formé. Les agents-robots ramassent les objets existants tout en consommant leur énergie pour rien. Plus le nombre d'objets augmente et plus les agents-robots ont des difficultés à bien faire, car leur énergie devient insuffisante, ce qui entraine une augmentation du nombre d'agents-robots morts.

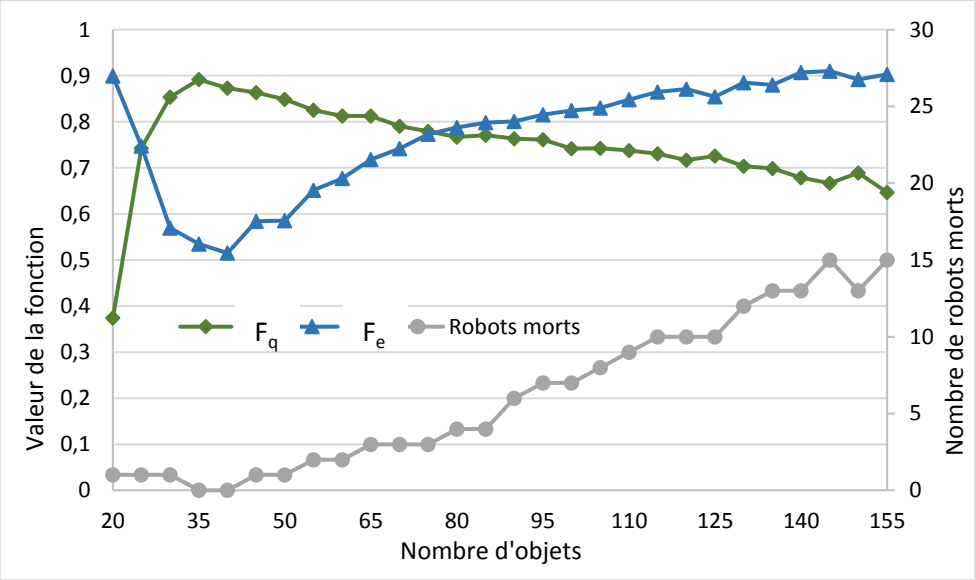


Figure 5.4 : L'effet du nombre d'objets

5.6.2. L'effet de la taille de l'environnement

Pour cette expérience, le nombre d'agents-robots et d'objets est fixe (nombre d'agents-robots = 30 et nombre d'objets = 60) et la dimension du terrain d'entraînement est variable. La figure 5.5 montre que dans un petit terrain d'entraînement, les mouvements des agents-robots sont limités ; bloqués entre les objets, les agents-robots changent continuellement leur perception locale de l'environnement sans déplacement réel, tout en espérant dépasser cet état de blocage. Dans le cas où le terrain d'entraînement devient plus grand, les agents-robots ont des difficultés à bien se comporter ; ils consomment plus d'énergie en parcourant de plus longues distances.

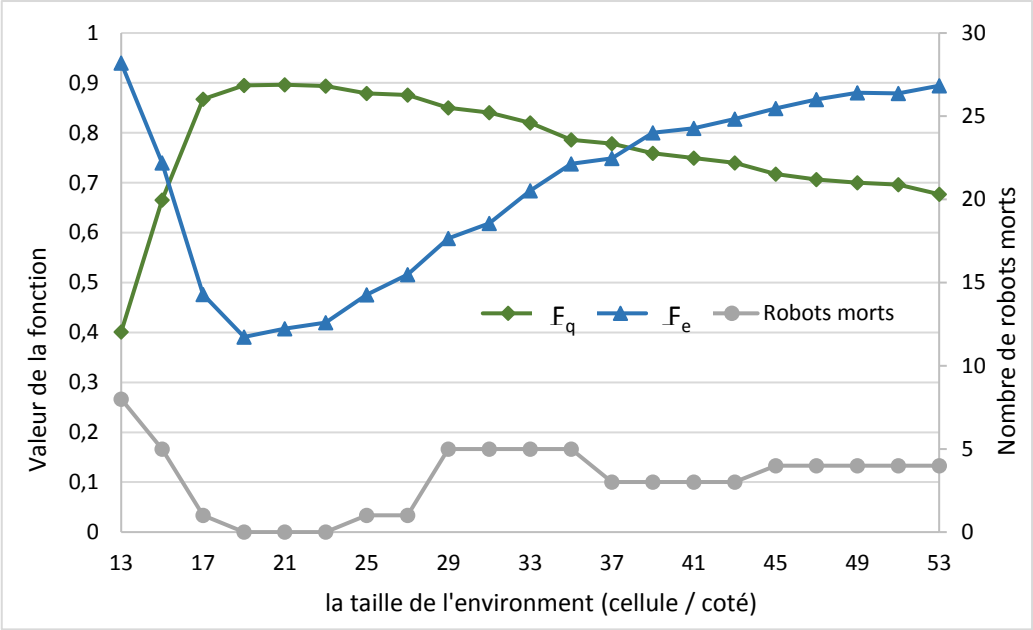


Figure 5.5 : L'effet de la taille de l'environnement

5.6.3. L'effet de la taille du problème

Pour cette expérience, la taille du terrain d'entraînement et le nombre d'agents-robots et d'objets sont varié pour obtenir le même problème dans différentes tailles en maintenant la densité des agents-robots et des objets constante, par rapport à la taille de l'environnement. Le tableau 5.2 résume les détails de chaque taille de problème, où il existe 2 objets pour chaque agent-robot. Les dimensions du terrain d'entraînement sont définies en fonction du nombre d'agents-robots utilisés dans chaque problème et de la densité des agents-robots dans le problème initial, calculée par $D = {33 * 33}/{30} = 36.3 \approx (6 * 6)$ cellules pour chaque agent-robot. Les résultats présentés dans la figure 5.6 montrent que les agents-robots sont plus performants dans les petits problèmes. Pour les problèmes plus importants, même si la densité du problème d'origine reste constante, les agents-robots doivent parcourir de longues distances pour atteindre la zone de regroupement. Ainsi, la tâche de regroupement est partiellement réalisée avec une capacité de batterie BC = 18000 unités.

Tableau 5.2 : Les tailles du problème

Référence de la taille du problème	**Nombre d'agents-robots**	**Nombre d'objets**	**Taille de l'environnement**
1	10	20	19x19
2	20	40	27x27
3	30	60	33x33
4	40	80	38x38
5	50	100	43x43
6	60	120	47x47
7	70	140	50x50
8	80	160	54x54
9	90	180	57x57
10	100	200	60x60

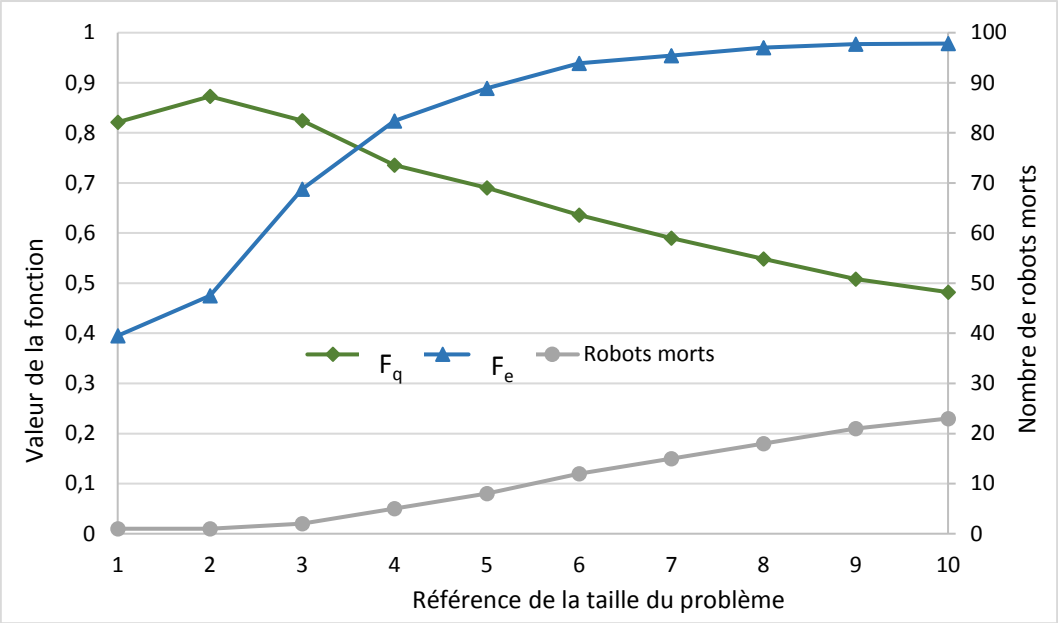


Figure 5.6 : L'effet de la taille du problème

5.7. Les facteurs dynamiques

Le modèle proposé, dans ce travail, contient de nombreux facteurs dynamiques, qui peuvent être présentés comme suit :

5.7.1. Les facteurs liés à l'essaim

Les principaux facteurs dynamiques qui sont à l'origine de la dynamique de l'essaim étudié sont : l'ajout de nouveaux agents-robots, la défaillance d'autres, le nombre d'agents-robots/objets et les changements de leurs positions. La nature totalement décentralisée du système robotique étudié et la connaissance limitée des agents-robots permettent de continuer à fonctionner sans qu'il soit nécessaire de coordonner, ce qui garantit une grande robustesse face à ces facteurs dynamiques.

5.7.2. Les facteurs liés aux caractéristiques des agents-robots

Par exemple la vitesse, le nombre et le type de capteurs/effecteurs. Ces facteurs dynamiques ne sont pas pris en compte dans ce travail. Dans les futurs travaux, la vitesse, par exemple, participera dans l'économie d'énergie. Sachant que plus de vitesse nécessite plus d'énergie, une des stratégies possibles sera : « lorsque le niveau d'énergie d'une batterie donnée atteint un certain seuil, l'agent-robot concerné diminuera sa vitesse pour consommer moins d'énergie, ce qui pourrait prolonger sa durée de vie ». Un nombre croissant de capteurs ou d'effecteurs peut respectivement augmenter la région de perception, ce qui entraine une augmentation du nombre de règles et un accroissement du nombre de comportements de base. La capacité de la batterie (CB) est une autre caractéristique des agents-robots qui peut avoir un effet significatif sur les performances du système. Pour les trois niveaux significatifs de la CB [18000, 15000, 12000], le système reste très robuste même si ses performances dégradent avec une capacité de batterie moins importante. Pour plus de 18000 unités, la même performance est obtenue, et pour moins de 12000 unités, la performance continue à se dégrader.

5.7.3. Les facteurs liés aux caractéristiques des objets et/ou les caractéristiques du sol

Par exemple, le poids et la forme des objets sont censés être adaptés aux capacités des agents-robots, et l'état du sol est censé être idéal et sans changement. Alors que l'ajout d'un grand nombre d'objets ou une extension importante du terrain d'entraînement peuvent affecter les performances du système.

5.8. Les facteurs accidentels

Le monde étudié dans ce travail est composé de trois éléments et peut être formulé comme suit : Monde = <Agents-Robots, Objets, Terrain>. Les éventuels facteurs accidentels peuvent être liés à une, deux ou trois de ces composantes :

5.8.1. Les facteurs liés aux agents-robots

A- **Batterie déchargée** : son effet est traité dans le travail présenté. Il convient de noter qu'une charge suffisante des batteries ne garantit pas toujours l'accomplissement de la tâche visée ; une batterie peut être accidentellement déchargée, pour une raison ou une autre. Inspirée de la gestion des stocks dans le domaine de la gestion industrielle, une marge de charge de sécurité supplémentaire peut réduire l'influence de ce facteur accidentel. Comme le montre la figure 5.7, un agent-robot consomme une énergie moyenne entre 15500 et 16500 unités. La capacité initiale utilisée est de 18000 avec une charge de sécurité supplémentaire de plus de 2500.

B- **Les pannes de capteurs/effecteurs** sont considérées comme des défaillances non tolérées dans ce travail. Un agent-robot souffrant de telles défaillances est considéré comme hors service. L'effet de ce facteur accidentel sur le système est le même qu'en (A) ; les performances du système se dégradent mais il reste robuste.

C- **Positions initiales des agents-robots** : Pour éviter toute distribution accidentelle des agents-robots, conduisant à des résultats expérimentaux meilleurs ou pires. Leurs positions initiales sont modifiées plusieurs fois (exactement 500 fois).

D- **Piégeage dans un état de blocage** : Parmi les stratégies de choix du lieu de déplacement ou de l'objet à manipuler, la plus déterministe consiste à spécifier le choix. Le robot est bloqué dans une situation où il ne peut pas fonctionner et il ne peut pas sortir. L'approche proposée donne plus de dynamisme dans ce sens. Pour exécuter le comportement sélectionné, le robot a deux choix (gauche ou droite) et celui qui est disponible sera exécuté. Si les deux choix ne sont pas disponibles, le robot peut sortir de cette situation de blocage en tournant à 180°. Si les deux choix sont disponibles, le robot choisit l'un d'entre eux.

5.8.2. Les facteurs liés aux objets

A- **Le poids et la forme** des objets sont considérés comme standard et les agents-robots peuvent toujours les manipuler.

B- **La répartition initiale des objets** : La fonction aléatoire utilisée peut générer des positions voisines formant de petits groupes dès le début de la simulation, en particulier dans le cas d'un grand nombre d'objets. Cette distribution accidentelle favorise la formation du tas et peut même conduire à une évaluation inappropriée du modèle comportemental, si elle est répétée plusieurs fois. La distribution initiale est modifiée plusieurs fois (exactement 500 fois) pour réduire la probabilité de rencontrer une telle distribution accidentelle.

5.8.3. Les facteurs liés à l'environnement

L'état du terrain d'entraînement est considéré comme idéal, non accidentel et invariant. Ces considérations seront prises en compte, à l'avenir, lors de la phase de passage au réel.

5.9. L'effet des facteurs dynamiques et accidentels

Pour montrer comment le système peut réagir à certains facteurs dynamiques ou accidentels, plusieurs scénarios ont été imaginés. Le scénario théorique (T), est un scénario où aucune perturbation

ne se produit pendant la simulation. Le scénario (A) relatif à l'ajout d'agents-robots a été réalisé, où trois groupes de cinq agents-robots (A-Rs) sont ajoutés au système à trois moments différents : 1500, 3000 et 4500. Un autre scénario (B) relatif au déplacement accidentel d'objets a été réalisé, dans lequel les positions de cinq objets sont modifiées de manière aléatoire. Ce phénomène est répété trois fois aux mêmes instants utilisés en (A), pour simuler l'effet d'un facteur externe qui perturbe la mission (le vent par exemple). Le dernier scénario (C) concernait les défaillances de capteurs ou d'effecteurs (considérées ici comme non tolérées) ou la décharge accidentelle de batteries. Trois agents-robots sont considérés comme hors service en même temps dans les trois moments susmentionnés. Notez que pour une comparaison équitable, les scénarios sont exécutés dans les mêmes conditions (5000 pas de temps pour chaque simulation et les simulations sont répétées 500 fois avec des conditions initiales différentes à chacune d'entre elles). Les résultats sont présentés dans le tableau 5.3.

Tableau 5.3 : L'effet de différents facteurs accidentels et dynamiques sur le système

		Scénario (T)	Scénario (A)	Scénario (B)	Scénario (C)
CB = 18000	F_q F_e A-Rs morts Robuste jusqu'à	0.80 0.88 0/30 la fin	0.75 0.70 0/45 la fin	0.73 0.88 0/30 la fin	0.73 0.89 9/30 la fin
CB = 15000	F_q F_e A-Rs morts Robuste jusqu'à	0.77 0.99 28/30 26 A-Rs morts	0.76 0.80 28/45 la fin	0.67 0.99 28/30 26 A-Rs morts	0.72 0.99 30/30 26 A-Rs morts
CB = 12000	F_q F_e A-Rs morts Robuste jusqu'à	0.75 0.99 30/30 27 A-Rs morts	0.72 0.82 30/45 la fin	0.59 0.99 30/30 27 A-Rs morts	0.70 0.99 30/30 27 A-Rs morts

- Le système reste très robuste en cas d'existence de changements, même si ses performances se dégradent.

- Dans le scénario (A), on peut distinguer deux cas différents : Le premier cas est celui où il n'y a pas besoin de nouveaux robots-agents pour succéder à la formation du tas ; les capacités des batteries sont suffisantes (CB = 18000). Elles restent donc toutes utiles jusqu'à l'épuisement des 5000 pas de temps. Dans le second cas, où les capacités des batteries sont de 15 000 ou 12 000, l'ajout de nouveaux robots-agents peut renforcer le système pour faire mieux ; les nouveaux agents-robots aideront à couvrir leurs collègues morts. L'ajout de nouveaux agents-robots, dans le premier cas, diminue la qualité de la formation du tas en raison de l'augmentation des interférences entre ces agents-robots. Dans le second cas, comme imprévu, le système fonctionne mieux dans le scénario (T) que dans ce scénario, ce qui prouve que les interférences ont un effet plus négatif sur les performances du système que le phénomène de mort des agents-robots.

- Le scénario (B) est celui qui a le plus d'effet sur les performances, sachant que la dégradation de la qualité des clusters devient plus importante avec un CB plus faible.

- Contrairement au scénario (T), où le phénomène des agents-robots morts ne peut apparaitre que dans les étapes ultérieures de la simulation, leur apparition dans les étapes antérieures de la simulation,

dans le scénario (C), a plus d'effet sur les performances du système robotique. Dans de tels cas, l'utilisation d'une CB élevée prolongera la durée de vie des agents-robots restant actifs pour combler le vide laissé par les agents morts.

5.10. Comparaison entre le modèle comportemental bi-objectif et mono-objectif

La comparaison entre les modèles comportementaux bi-objectif et les modèles comportementaux à objectif unique obtenus dans d'autres travaux est inappropriée, car ils ne sont pas développés dans les mêmes conditions. Ainsi, un algorithme génétique ordinaire a été mis en œuvre, pour récupérer l'un des meilleurs modèles comportementaux à objectif unique (MC-MO), réalisé dans les mêmes conditions que le MC-BO. Les expériences sont réalisées en utilisant le même nombre d'agents-robots (30) et d'objets (60) et la même dimension de terrain d'entraînement (33x33). Les agents-robots ont la même région de perception et les mêmes 3 comportements de base. Contrairement à l'approche proposée, il n'y a pas de distinction entre les niveaux d'énergie ; le mouvement est codifié par {0}, les comportements de ramassage et de dépôt sont codifiés par {1} et le choix de la cellule est fait de manière aléatoire. L'algorithme génétique à objectif unique susmentionné, avec les mêmes paramètres, est appliqué et la meilleure performance obtenue en termes de qualité est $F_q = 0.728$.

Contrairement à l'approche adoptée, aucune information n'est disponible sur l'énergie consommée, au cours de ce processus de conception d'un modèle comportemental à objectif unique. Après comparaison de MC-MO avec MC-BO, les observations suivantes ont été faites. Premièrement, malgré le fait que le MC-BO offre une meilleure qualité que le MC-MO, il consomme également moins d'énergie. La figure 5.7 montre la consommation moyenne d'énergie de chaque agent-robot, en utilisant ces deux modèles comportementaux, sur plus de 500 simulations avec des conditions initiales aléatoires, pendant 5000 pas. En bref, le MC-BO est plus performant sur les deux objectifs. Deuxièmement, la figure 5.7 montre qu'en utilisant le MC-MO, la consommation d'énergie est plus perturbée, d'une simulation à l'autre. La variance, sur 500 simulations de l'énergie consommée de MC-BO, est de 73230,64, alors que la variance de MC-MO est de 149149,8, soit plus du double. Cela confirme que l'énergie consommée, en utilisant le MC-MO, est plus perturbée et moins rationalisée d'une expérience à l'autre, car elle n'est pas contrôlée ; plusieurs actions qui consomment une quantité différente d'énergie sont considérées comme équivalentes.

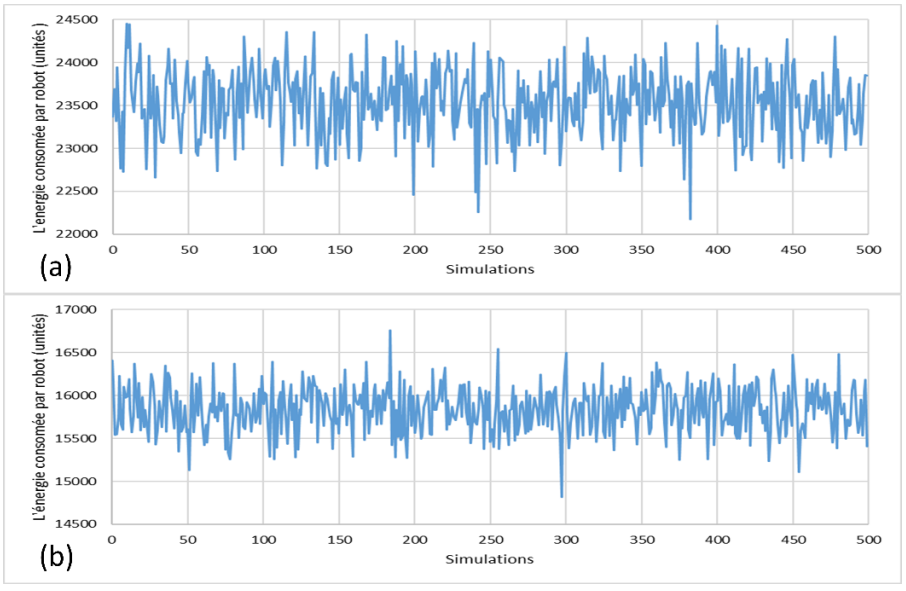


Figure 5.7 : L'énergie consommée par les agents-robots en utilisant (a) MC-MO (b) MC-BO

Tableau 5.4 : L'effet de la capacité de batteries sur le système utilisant différents modèles comportementaux

			MC-MO	MC-BO	MC-BO'
L'effet de la capacité de batteries et la robustesse	**CB = 27000**	F_q F_e A-Rs morts Robuste jusqu'à	0.67 0.87 0/30 la fin	0.80 0.58 0/30 la fin	0.66 0.49 0/30 la fin
	CB = 18000	F_q F_e A-Rs morts Robuste jusqu'à	0.62 0.99 30/30 25 A-Rs morts	0.80 0.88 0/30 la fin	0.67 0.74 0/30 la fin
	CB = 15000	F_q F_e A-Rs morts Robuste jusqu'à	0.58 1 30/30 26 A-Rs morts	0.77 0.99 28/30 26 A-Rs morts	0.67 0.89 1/30 la fin
	CB= 12000	F_q F_e A-Rs morts Robuste jusqu'à	0.52 1 30/30 26 A-Rs morts	0.75 0.99 30/30 27 A-Rs morts	0.65 0.99 25/30 24 A-Rs morts

Les tableaux 5.4 et 5.5 résument la comparaison effectuée entre MC-MO, MC-BO et un autre modèle comportemental bi-objectif, appelé MC-BO' ayant la même qualité de regroupement que MC-MO. MC-BO' est comparé à MC-MO, pour prouver que même si deux modèles comportementaux, ayant une énergie suffisante pour atteindre les mêmes performances en termes de qualité de regroupement, l'approche bi-objective surpasse l'approche mono-objective, lorsque l'énergie de la batterie devient limitée, comme dans les applications du monde réel. Les expériences sont réalisées avec les mêmes paramètres et conditions que ceux présentés dans les sections précédentes. Les résultats surlignés dans le tableau 5.4 sont les meilleurs.

D'après le tableau 5.4, nous pouvons décrire que :

- Avec une batterie de grande capacité, les trois modèles comportementaux fonctionnent normalement, et tous les agents-robots restent fonctionnels jusqu'à la fin. Avec des batteries de plus faible capacité, le système utilisant le MC-MO commence à perdre des membres plus tôt que le MC-BO et le MC-BO' ;

- On obtient une meilleure qualité avec des capacités de batterie plus faibles, en utilisant l'un des deux modèles comportementaux bi-objectif analysés (MC-BO ou MC-BO') ;

- Comme le système est décentralisé et homogène, les trois modèles comportementaux sont robustes tant que 3 à 6 parmi les 30 agents-robots sont encore actifs ;

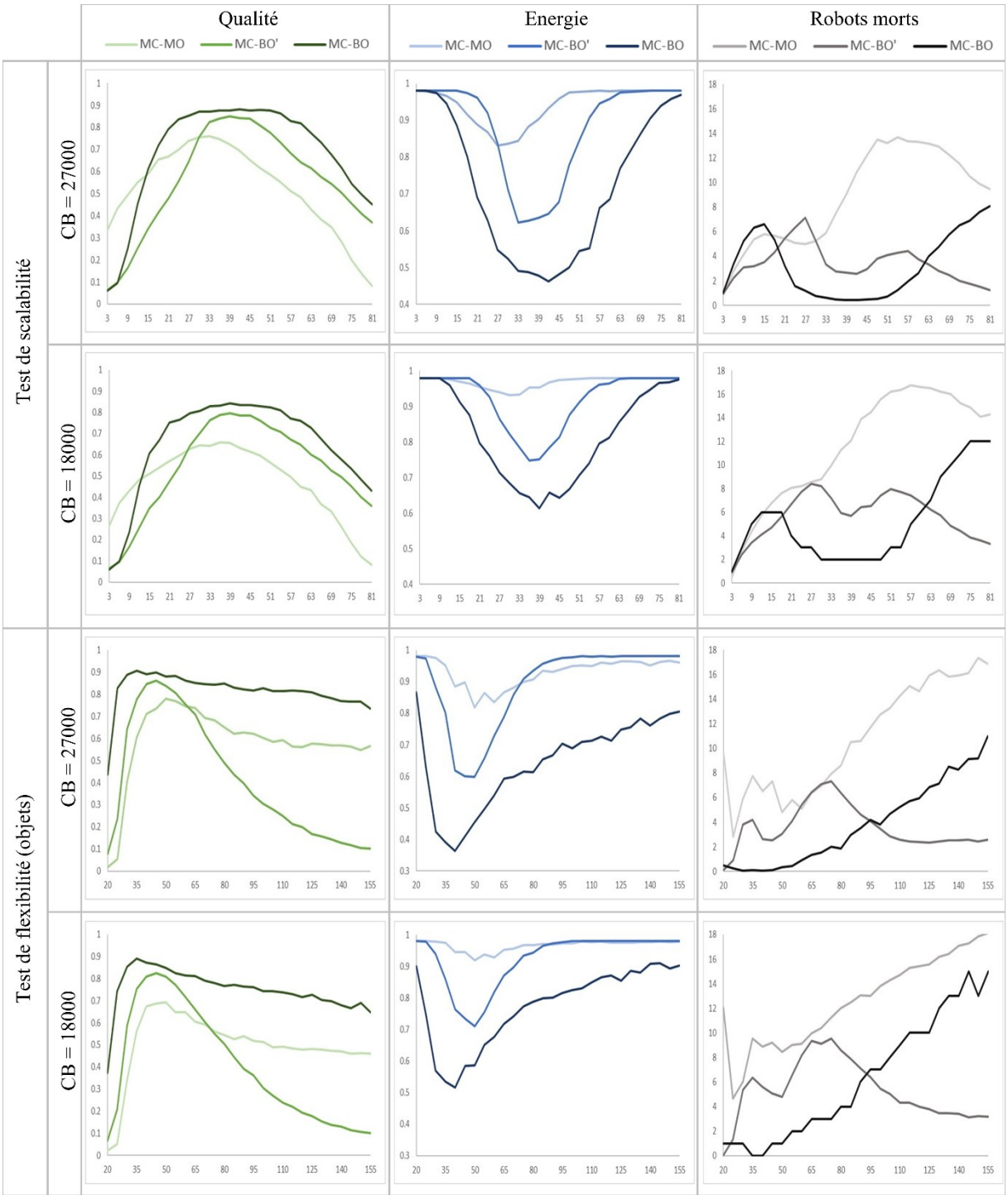


Figure 5.8 : Comparaisons entre les différents modèles comportementaux selon la scalabilité et la flexibilité

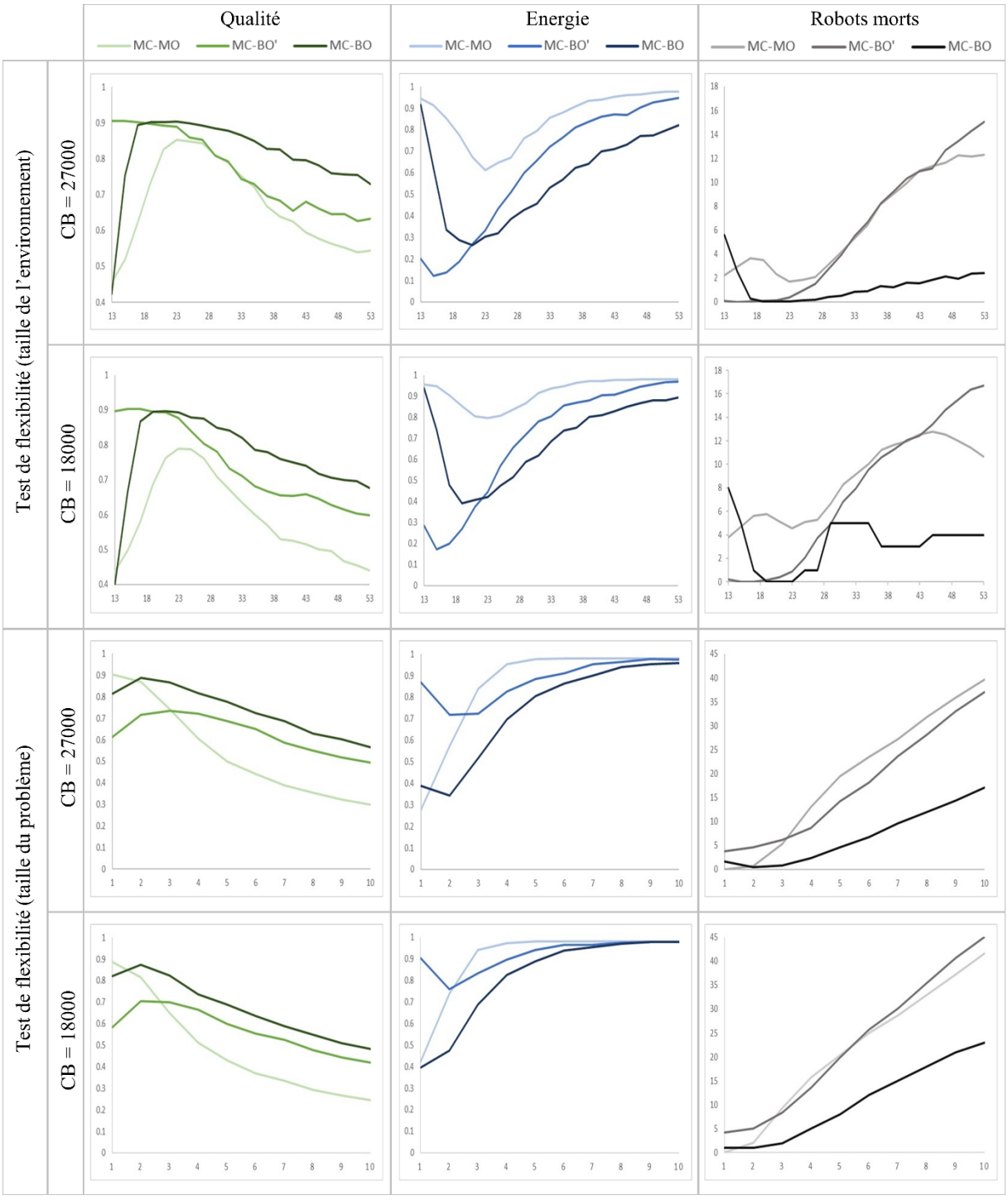


Figure 5.8 : continuée

D'après la figure 5.8, nous pouvons décrire que :

- Dans les expériences de scalabilité et de flexibilité, le MC-BO consomme moins d'énergie que le MC-BO', car il atteint le seuil fixé de la qualité (0,9) plus rapidement que le MC-BO', avec moins d'agents-robots morts dans la plupart des expériences ;

- Pour tous ces modèles comportementaux, le système robotique est plus scalable et plus flexible avec des batteries de grande capacité qu'avec des batteries de faible capacité ;

- Le MC-MO consomme plus d'énergie dans toutes les expériences et compte plus d'agents-robots morts dans la plupart d'entre elles ;

- Le système robotique est plus scalable en utilisant MC-BO ou MC-BO' qu'en utilisant MC-MO ;

- Le système robotique est plus flexible en utilisant MC-BO qu'en utilisant MC-MO dans tous les cas ;

- Le système robotique est plus flexible en utilisant MC-BO' qu'en utilisant MC-MO dans les expériences relatives aux dimensions du terrain d'entraînement et à la taille du problème.

5.11. Conclusion

Dans ce chapitre, une étude expérimentale est faite afin d'examiner la robustesse, la scalabilité et la flexibilité du système. D'autres expériences sont faites pour étudier la manière dont le système réagit face aux différents facteurs dynamiques et accidentels. Aussi une comparaison est faite entre les modèles comportementaux bi-objectif et mono-objectif, pour prouver la supériorité de l'approche proposée. Les résultats montrent que le système robotique reste robuste tant que plus de 14 % des agents-robots sont encore actifs. La solution bi-objectif est plus adaptée, car elle permet d'économiser de l'énergie tout en offrant une meilleure qualité. De plus, elle est plus scalable et plus flexible qu'une solution à objectif unique.

Chapitre 6: L'algorithme d'optimisation basé sur l'enseignement-apprentissage pour les problèmes multi-objectif discrets à grande échelle

6.1. Introduction

De nombreuses études ont été réalisées pour résoudre le problème de regroupement d'objets sous sa forme mono-objectif, comme [4, 6, 196, 280]. Dans le chapitre 4, nous avons remodelé le problème de regroupement d'objets conventionnel en un autre problème bi-objectif, où les modèles comportementaux de compromis, permettant de maximiser la qualité du regroupement et de minimiser l'énergie consommée par les robots, sont sujets de la recherche évolutive [9]. Nous avons découvert qu'il existe une relation étroite entre le choix des comportements, la qualité du regroupement et l'énergie consommée. Afin d'améliorer la qualité du front de Pareto obtenu par NSGA-II, une variante de l'algorithme basé sur l'enseignement-apprentissage est proposée dans ce chapitre.

En plus de paramètres communs, l'algorithme NSGA-II, comme d'autres algorithmes basés sur la population, nécessite des paramètres spécifiques à l'algorithme tel que la probabilité de mutation et de croisement. Le réglage de ces paramètres est un facteur très important qui influe sur les performances de l'algorithme [281]. Un mauvais réglage affecte la vitesse de convergence et peut mener la recherche vers des optimums locaux. Par contre, l'algorithme TLBO (Teaching-Learning based optimization) original et d'autres variantes ne nécessitent aucun paramètre spécifique. Cet algorithme inspiré de la société humaine est classé dans la nouvelle génération des algorithmes basés sur la population. Il est basé sur deux phases principales : la phase d'enseignement et la phase d'apprentissage.

Le reste de ce chapitre est organisé comme suit. La section 6.2 décrit la motivation et la taxonomie du travail présenté dans ce chapitre. Dans la section 6.3, l'algorithme proposé est présenté. Dans la section 6.4, les résultats obtenus en utilisant cet algorithme sur le problème de regroupement d'objets bi-objectif sont présentés et comparés à ceux de NSGA-II.

6.2. Motivation et taxonomie de ce travail

Le point commun entre la majorité des algorithmes à base de population est les paramètres spécifiques à chaque algorithme. Ces paramètres jouent un rôle important dans l'algorithme. L'ajustement de ces derniers affecte directement l'efficacité de l'algorithme. Tandis que l'optimisation basée sur le processus enseignement-apprentissage est une technique qui ne nécessite aucun réglage de paramètre spécifique de l'algorithme. Malgré que cet algorithme soit récent, il est devenu de plus en plus populaire grâce à son coût de calcul réduit et à sa grande efficacité. Plusieurs versions sont proposées et appliquées avec succès dans différents domaines. Les résultats montrent l'efficacité appréciable de l'algorithme TLBO pour résoudre les problèmes multi-dimensionnels, linéaires et non-linéaires. Depuis la proposition de sa première version par Rao et al. [2, 3], l'algorithme de base a été amélioré pour accroitre ses performances et ses capacités d'exploration et d'exploitation [8].

Comparé à d'autres algorithmes heuristiques, l'algorithme TLBO est simple, facile à décrire et à implémenter. Il n'a pas de paramètre spécifique. De plus, TLBO a une grande précision, une bonne performance de convergence et une bonne robustesse sur les différents problèmes d'optimisation. Ces dernières années, l'algorithme TLBO a été grandement amélioré et largement utilisé [106].

Les versions de TLBO proposées sont dédiées aux problèmes multi-objectif qui sont généralement continus et/ou à petite échelle. Une variante de TLBO a été proposée pour résoudre les problèmes mono-objectif à grande échelle [3]. Malgré que les résultats prouvent la capabilité de TLBO à résoudre les problèmes à grande échelle, aucune variante n'a été proposée pour résoudre les problèmes multi-objectif à grande échelle. La taxonomie de l'algorithme proposé est résumée dans la figure 6.1.

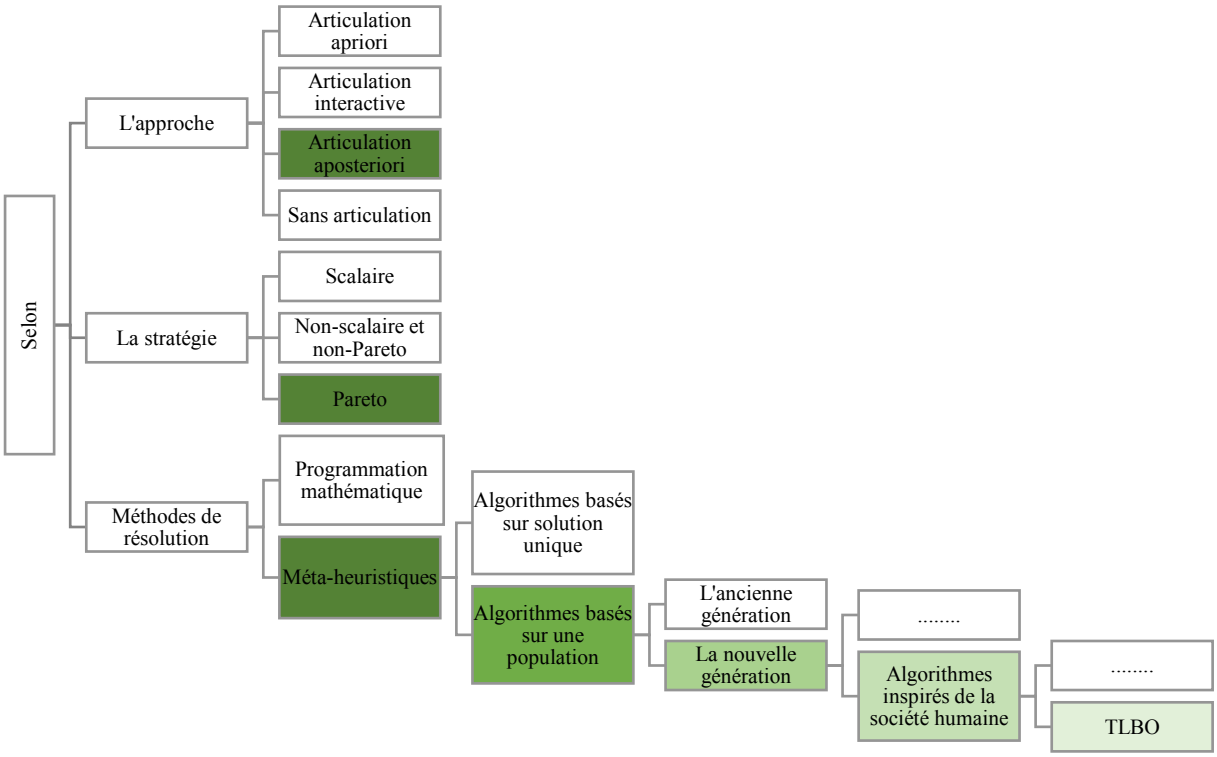


Figure 6.1 : Taxonomie de l'algorithme proposé

6.3. L'algorithme proposé DLM-TLBO

L'algorithme proposé, appelé Discrete Large-scale Multi-objective Teaching-Learning-Based Optimization (DLM-TLBO), commence avec N apprenants aléatoires. Les capacités des apprenants sont obtenues en calculant les valeurs des fonctions objectives. Ensuite, le mécanisme de tri non dominé (non-dominated sorting) de NSGA-II est adopté pour les classer comme il est décrit dans la figure 6.2. En se basant sur cette classification, chaque classe d'apprenants suivra un processus différent. Les nouveaux apprenants qui en résultent sont évalués. Ensuite, la nouvelle population et l'ancienne population sont fusionnées pour trier les individus en fonction du rang et de la distance d'encombrement. Les N meilleurs apprenants sont sélectionnés pour participer à la prochaine itération.

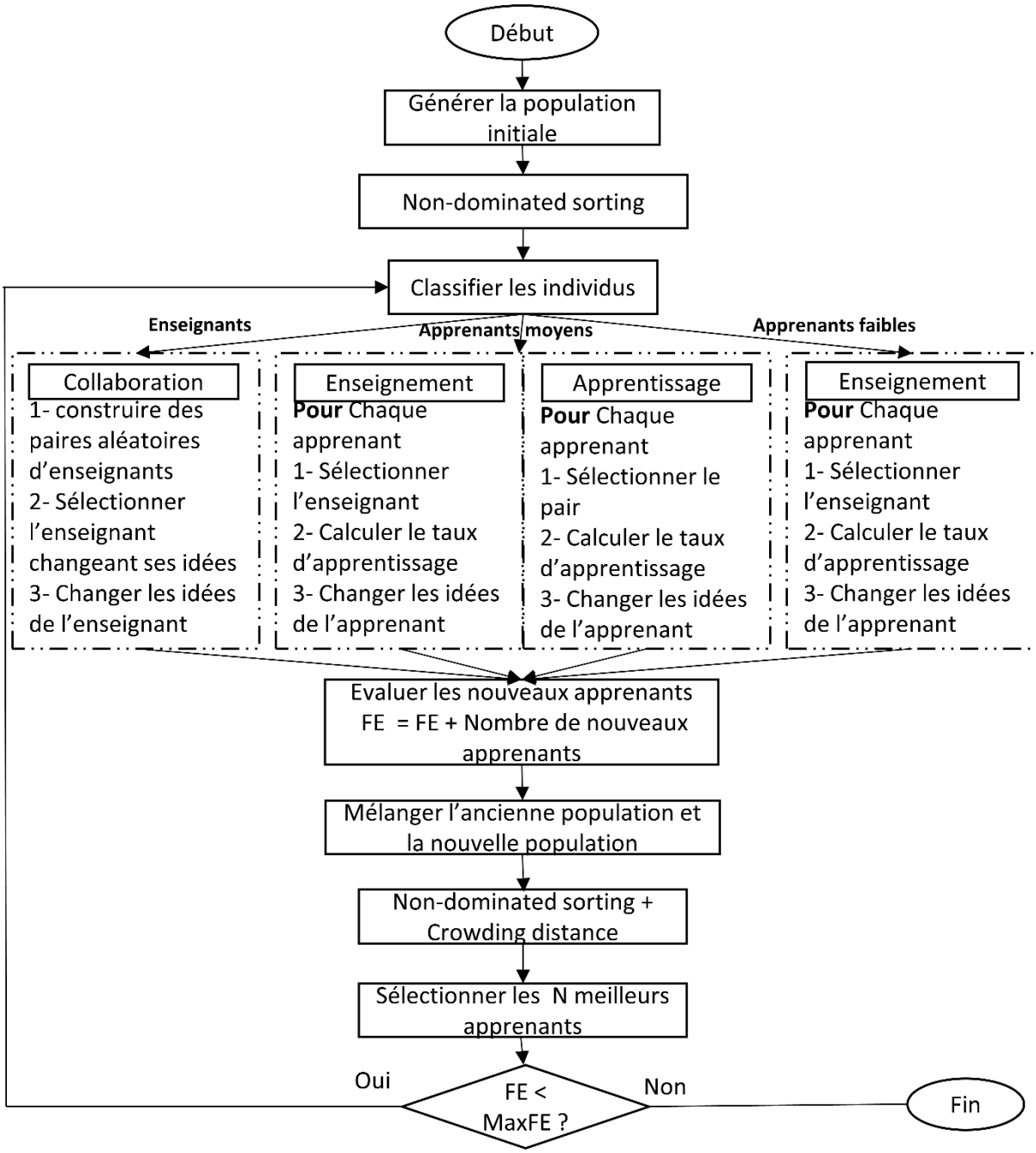


Figure 6.2 : L'organigramme de l'algorithme DLM-TLBO

6.3.1. La classification hiérarchique

En fonction de leur rang, les individus sont classés en : enseignants avec le rang égal à 1, apprenants faibles (débutants) ayant le rang maximal et apprenants moyens avec $1 < rang < max(rang)$. Les enseignants choisissent une phase de collaboration ; les apprenants moyens participent aux deux phases d'enseignement et d'apprentissage ; et les apprenants faibles participent à la phase d'enseignement uniquement. Ce phénomène est similaire à la réalité où les débutants suivent leurs enseignants, les apprenants moyens suivent les enseignants et enrichissent leur bagage en contactant d'autres apprenants du niveau supérieur, tandis que les enseignants collaborent, par exemple en effectuant des travaux de recherche.

6.3.2. La sélection des enseignants et des pairs

La sélection de l'enseignant en fonction de la distance d'encombrement la plus élevée risque d'orienter les apprenants vers un optimum local [155]. Pour la raison que le problème traité est un problème à grande échelle, la diversité de la recherche doit être prise en considération. Pour améliorer la diversité de l'algorithme proposé, l'enseignant t d'un apprenant spécifique l est choisi au hasard parmi les individus du premier front. Différemment des autres versions de TLBO, où le pair est choisi au hasard parmi tous les individus de la population, le pair p dans la version proposée est choisi au hasard parmi les individus du front suivant : $rang(p) = rang(l) - 1$. Ce qui garantit le changement des idées de l'apprenant dans le bon sens, résultant par la suite à une amélioration de ses performances. La figure 6.3 résume la classification hiérarchique et la méthode de sélection des enseignants et des pairs.

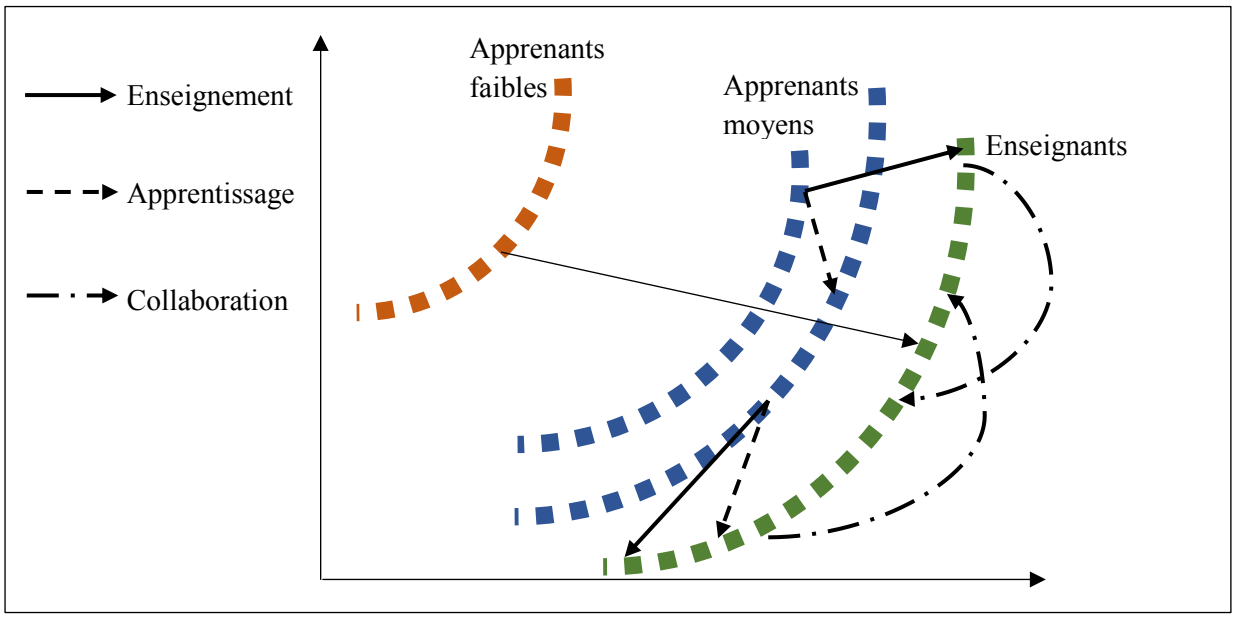


Figure 6.3 : Principe de fonctionnement de DLM-TLBO

6.3.3. Les phases d'enseignement et d'apprentissage modifiées

En raison de la nature discrète et à grande échelle du problème, nous proposons d'utiliser ce que nous pouvons appeler « le changement radical des idées », l'enseignant change certaines idées fausses (variables) de ses apprenants en les remplaçant par ses propres idées, en espérant que cela aidera l'apprenant à améliorer ses résultats. La même technique est utilisée dans la phase d'apprentissage. La perturbation du transfert de connaissances est garantie dans d'autres variantes de l'algorithme en utilisant la moyenne de la population. Dans la nôtre, les connaissances sont transférées individuellement et la perturbation est assurée par le choix aléatoire des variables à remplacer (voir la figure 6.4).

6.3.4. La phase de collaboration

Les enseignants, dans le premier front, participeront à la phase de collaboration. Les enseignants sont répartis au hasard par paires (t_1, t_2). L'enseignant qui changera ses idées est choisi au hasard et un nouvel individu est obtenu en utilisant le même processus décrit dans les phases d'enseignement et d'apprentissage et présenté sur la figure 6.4.

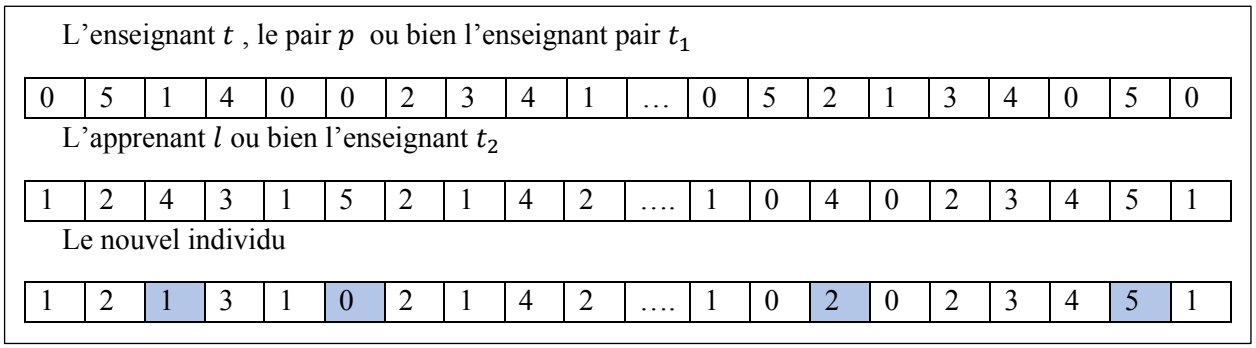


Figure 6.4 : Le principe de changement d'idées

6.3.5. Taux d'apprentissage adaptatif

Dans l'algorithme original, l'individu apprend tout de son enseignant si $T_f = 2$ et rien si $T_f = 1$. Dans l'algorithme proposé, et suivant d'autres versions améliorées, le taux d'apprentissage utilisé doit varier dans un intervalle pour simuler le processus réel, où les apprenants peuvent apprendre avec proportions différentes. Aussi dans l'algorithme original, les connaissances acquises par l'apprenant sont affectées par les capacités des autres apprenants, en utilisant la moyenne du groupe. Cela signifie que tous ces apprenants peuvent converger vers la même zone à cause d'avoir le même enseignant. De ce fait, la distribution de l'ensemble sera diminuée. Dans l'algorithme proposé, les connaissances acquises dépendent uniquement des capacités individuelles de l'apprenant et de l'enseignant, en utilisant le taux d'apprentissage adaptatif LR inspiré du facteur d'enseignement adaptatif TF calculé dans l'espace objectif [105]. À la différence, ce taux définira le pourcentage de changements dans les idées de l'apprenant (variables). Le taux d'apprentissage adaptatif $\in [0, 1]$ est calculé, à l'aide de l'équation (6.1), entre l'apprenant l et son enseignant t, l'apprenant l et son pair p ou bien entre-deux enseignants t_1 et t_2. Où, M est le nombre d'objectifs.

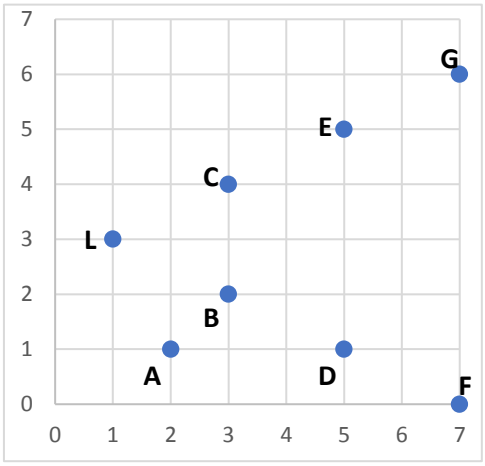
$$LR = \frac{1}{M} \sum_{m=1}^{M} \frac{min\{f_m^l, f_m^t\}}{max\{f_m^l, f_m^t\}} \qquad (6.1)$$

Ce taux est calculé dans l'espace objectif et non pas dans l'espace de décision pour deux raisons : (1) la distance entre les individus dans l'espace de décision ne donne pas une vraie estimation de la différence entre leurs performances à cause de l'inexistence d'une relation claire et nette entre les valeurs de variables au niveau micro et les fitness obtenues au niveau macro ; (2) le calcul de la distance dans un espace à grande échelle devient coûteux en temps de calcul.

L'effet de la distance entre deux individus sur la valeur de LR est étudié, en choisissant 7 cas. Comme le montre le tableau 6.1, si les individus sont loin l'un de l'autre, le taux LR sera petit et vice-versa. Dans [98], l'enseignant d'un certain apprenant est le plus proche parmi ceux qui le dominent. Dans l'algorithme proposé, le choix de l'enseignant/pair reste aléatoire, mais l'individu apprend mieux auprès d'un individu avec un niveau proche de son niveau. L'apprenant L apprend mieux si son enseignant/pair est A, B ou bien C et il apprend moins dans le cas de D, E, F ou G.

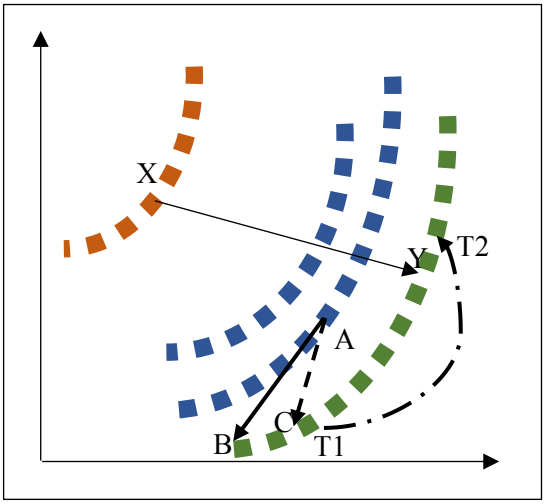
La valeur de LR n'est pas seulement affectée par la distance entre les individus mais aussi affectée par la direction. Comme le montre le tableau 6.1, la distance euclidienne (DE) entre l'apprenant L et les candidats A, B et C est la même, alors que LR est différent. Malgré que la distance euclidienne DE (L, D) et DE (L, E) est la même la diminution de LR est plus importante dans le cas de D par rapport au cas de E (ce qui est le cas de F et G aussi). Aussi dans le tableau 6.2, malgré que la distance entre X et Y est plus importante que celle entre A et B, le taux LR (X, Y) est supérieur à LR (A, B). Il est observé, en utilisant NSGA-II que la zone supérieure du front de Pareto est plus difficile à explorer et le nombre de solutions trouvées est moins que celles trouvées dans la zone inférieure. Ce taux d'apprentissage est utilisé pour favoriser la recherche dans cette zone supérieure.

Tableau 6.1 : Effet de la distance et la direction sur le taux d'apprentissage adaptatif



L'apprenant (F_q, F_e)	L (1,3)	**DE**	**LR**
L'enseignant/pair possible (F_q, F_e)	A (2,1)	2.23	0.41
	B (3,2)	2.23	0.5
	C (3,4)	2.23	0.54
	D (5,1)	4.47	0.26
	E (5,5)	4.47	0.4
	F (7,0)	6.7	0.07
	G (7,6)	6.7	0.32

Tableau 6.2 : Calcul du taux d'apprentissage adaptatif dans différents cas



L'apprenant / L'enseignant (F_q, F_e)	**Son enseignant /pair** (F_q, F_e)	**DE**	**LR**
X (0.3,0.6)	Y (0.8,0.4)	0.53	0.52
A (0.6,0.3)	B (0.45,0.03)	0.3	0.42
	C (0.55,0.06)	0.24	0.55
T1 (0.6,0.07)	T2 (0.82,0.5)	0.48	0.43

6.4. Résultats de DLM-TLBO

Les performances de l'algorithme proposé sont testées sur le problème de regroupement d'objets décrit dans le chapitre 4. Le fait qu'on doit trouver 486 comportements, le problème décrit est classé parmi les problèmes à grande échelle. Les résultats obtenus par cet algorithme sont aussi comparés avec ceux de NSGA-II. Pour une comparaison équitable, les deux algorithmes sont exécutés avec les mêmes paramètres. Aussi, l'exécution de chaque algorithme est répétée 10 fois, la moyenne de la population est calculée et enregistrée pour les deux fonctions objectifs durant les 10 exécutions, la meilleure approximation du front du Pareto sur les 10 est aussi présentée. Les individus du premier front de toutes les exécutions sont triés et la distance d'encombrement entre ces individus est calculée. Par la suite, seuls les individus non-dominés sont choisis pour représenter l'approximation du front obtenue par l'algorithme.

Les paramètres généraux sont fixés comme suit : le nombre maximum d'évaluations Max-FE = 30.000 est fixé comme condition d'arrêt, la taille de la population N = 65, aussi chaque individu est entrainé et évalué 10 fois (T = 10), en changeant les conditions initiales. Le temps d'entrainement est 2000 pas de temps. Alors que les paramètres du problème sont : le nombre d'agents-robots est 30, le nombre d'objets est 60 avec 9000 unités comme énergie initiale, la taille de l'environnement est 33x33 cellules. Tandis que DLM-TLBO ne nécessite aucun paramètre spécifique, NSGA-II nécessite la spécification des paramètres suivants : le pourcentage de croisement est 0.9 et le pourcentage de mutation est 0.4.

6.4.1. L'évaluation de performance sur le problème de regroupement

Pour étudier les performances de DLM-TLBO, l'algorithme décrit dans la section précédente est implémenté sous NetLogo, dans le but de résoudre le problème de regroupement bi-objectif présenté dans le chapitre 4, avec les paramètres déjà spécifiés. Après avoir répété l'exécution 10 fois, la procédure Non-Dominated Sorting est utilisée pour ranger toutes les solutions trouvées. La figure 6.5 représente les solutions non-dominées parmi toutes les solutions trouvées, alors que pour obtenir les 10, 20 ou 30 meilleures solutions, présentées sur la figure 6.5, les solutions non-dominées sont triées selon la distance d'encombrement.

Comme on l'a déjà mentionné, le front de Pareto réel du problème traité est inconnu, ce qui est le cas dans pas mal de problèmes d'optimisation réels. La seule information qu'on a c'est que la meilleure qualité est proche de ou égale à 1. Comme le présente la figure 6.5, l'approximation obtenue en utilisant l'algorithme DLM-TLBO est proche du front de Pareto réel inconnu, le fait que la meilleure qualité de regroupement obtenue est égale à 0.96. Dans la figure 6.6, les 30, 20 et 10 meilleures solutions obtenues sont aussi présentées afin de donner au décideur de multiple choix et simplifier la tâche de décision du modèle comportemental à intégrer dans les agents-robots.

Les profils de convergence, présentés par l'évolution de la moyenne de la qualité de formation du tas et la moyenne de la consommation d'énergie de toute la population, sont enregistrées est présentées sur la figure 6.6. La courbe de l'évolution de la moyenne de la qualité de formation se compose essentiellement de deux parties. Dans la première partie, l'évolution est plus rapide que dans la deuxième partie où la courbe devient plus stable. Tandis que la courbe de l'évolution de la moyenne de consommation de l'énergie se compose de trois parties. Dans la première partie, la consommation augmente en raison du nombre important de cas d'incompatibilité entre les perceptions et les comportements à exécuter, générés aléatoirement. Dans la deuxième partie, les efforts de l'algorithme pour optimiser les deux objectifs commencent à apparaitre, la moyenne de consommation diminue et celle de la qualité continue son augmentation. Alors que dans la dernière partie, la moyenne de consommation devient stable aussi.

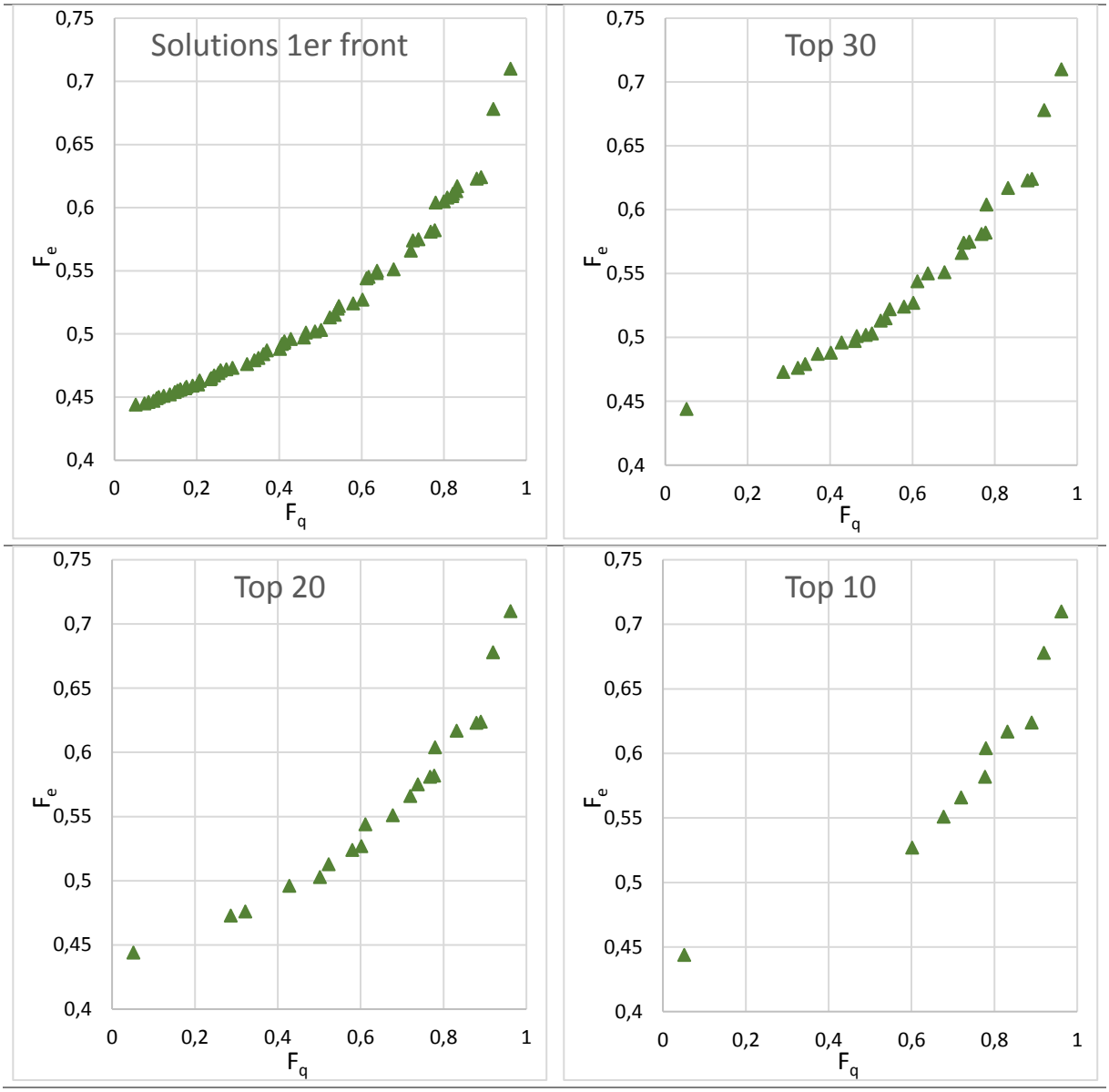


Figure 6.*5.* : L'approximation du front de Pareto obtenue par DLM-TLBO

Le modèle comportemental qui donne une qualité de formation du tas égale à 0.96 est utilisé pour prendre des instances du temps de la manière dont l'essaim réalise le regroupement d'objets bi-objectif. La figure 6.7 présente un aperçu chronologique pris dans différentes instances de temps. Malgré que la tâche de regroupement soit stochastique et le temps nécessaire pour accomplir la tâche peut être différent d'une exécution à une autre, on peut dire que la solution qui donne la meilleure qualité, générée par DLM-TLBO, prend moins de temps que celle générée par NSGA-II ; en plus du fait qu'elle génère une qualité de formation du tas plus élevée en consommant moins d'énergie.

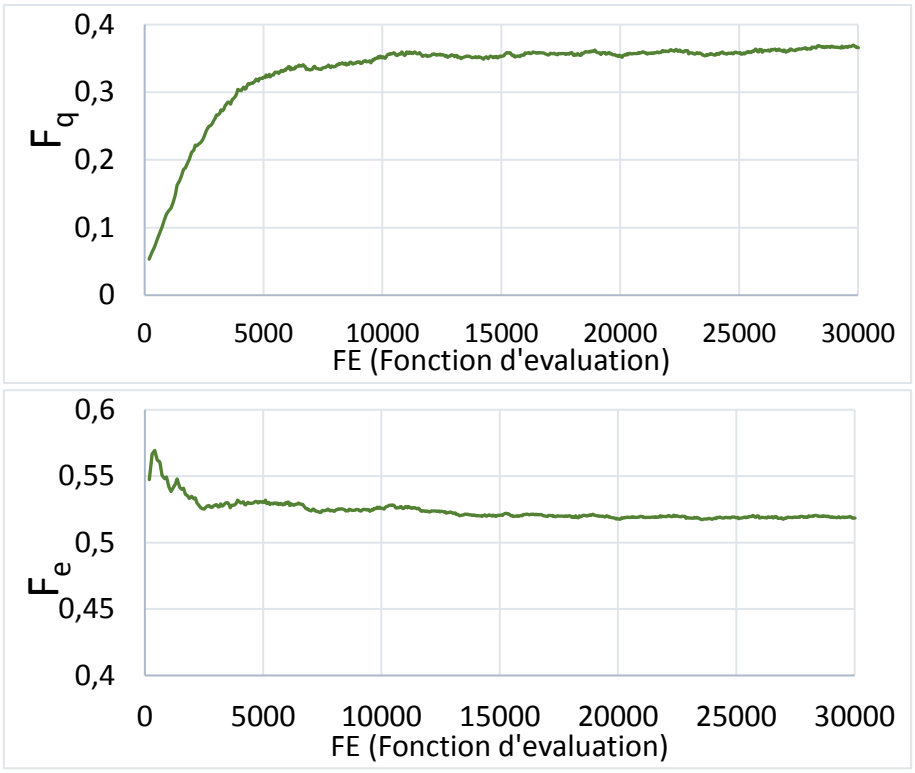


Figure 6.6 : Les profils de convergence des deux objectifs durant 10 exécutions en utilisant DLM-TLBO

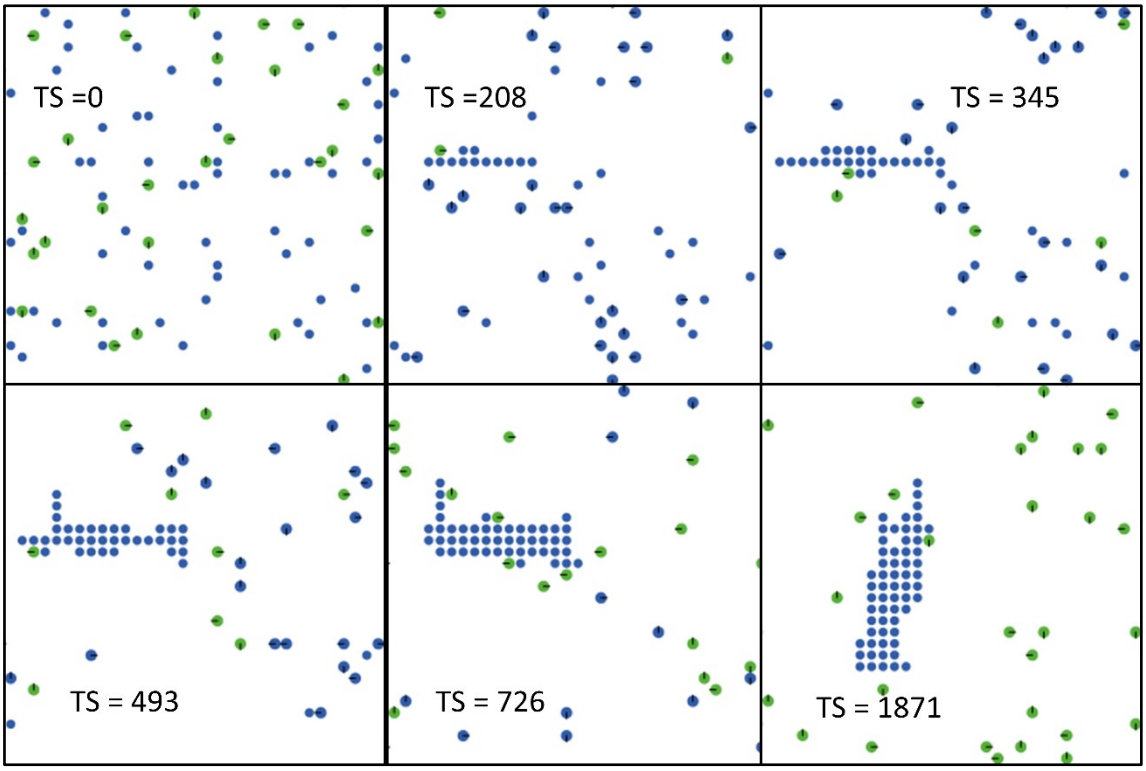


Figure 6.7 : Aperçu chronologique de la tâche de regroupement d'objets

6.4.2. Comparaison entre les résultats de NSGA-II et DLM-TLBO

Par la suite, les résultats obtenus en utilisant les deux algorithmes : NSGA-II et DLM-TLBO sont comparés. Dans un premier temps, les meilleures approximations du front de Pareto sont comparées, comme le présente la figure 6.8. En termes de convergence, le front généré par DLM-TLBO domine celui généré par NSGA-II. En termes de diversification, la partie inférieure du front, dans les deux algorithmes, est surpeuplée à cause de la nature du problème. Où il est plus difficile de trouver des solutions qui génèrent une bonne qualité que de trouver des solutions qui consomment moins d'énergie. Ces régions du front deviennent de moins en moins peuplées, en présentant les meilleures solutions selon la distance d'encombrement. L'utilisation du taux d'apprentissage LR résulte en une bonne convergence dans la zone supérieure du front.

Aussi les profils de convergence des deux objectifs durant les 10 exécutions sont enregistrés et la moyenne de ces exécutions est présentée sur la figure 6.9. Comme le montre la figure, l'algorithme proposé converge plus vite que NSGA-II. En plus, DLM-TLBO atteint des fitness moyennes plus importantes que celles atteintes par NSGA-II, dans les deux objectifs.

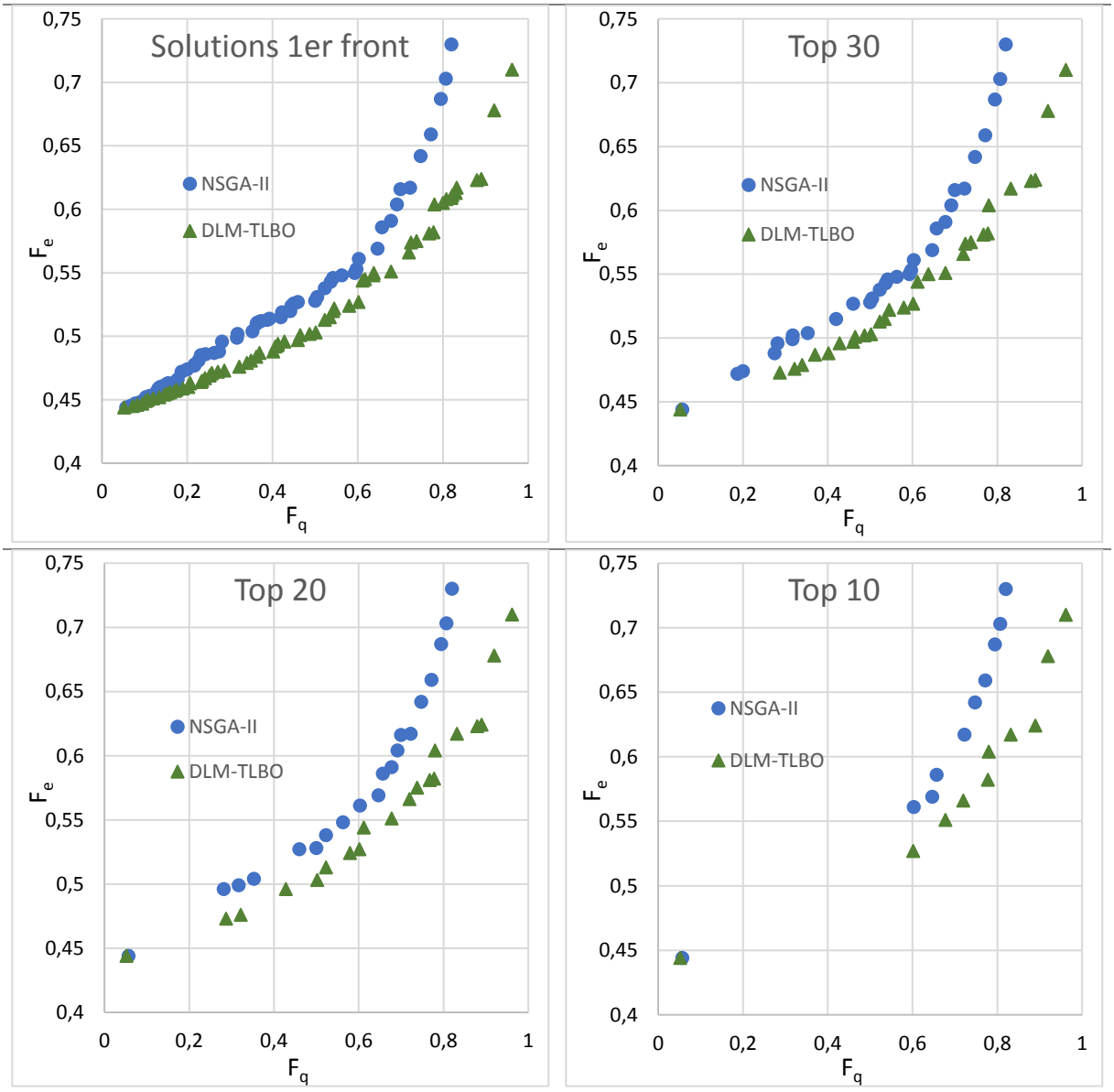


Figure 6.8 : Comparaison entre les approximations du front de Pareto obtenues par NSGA-II et DLM-TLBO

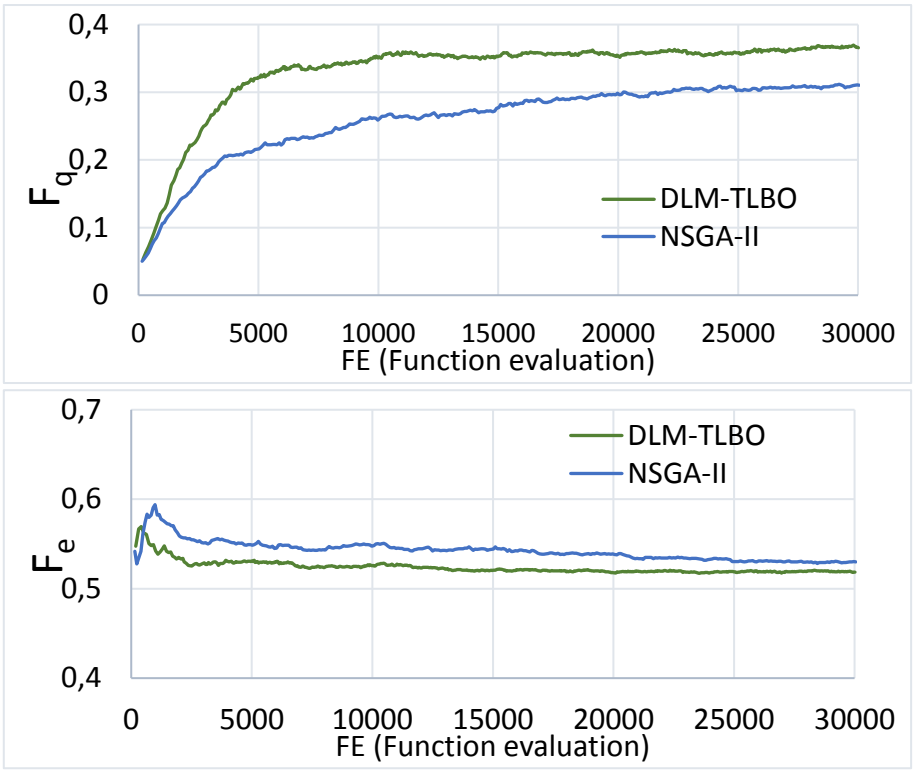


Figure 6.9 : Comparaison de la vitesse moyenne de convergence des deux objectifs durant 10 exécutions en utilisant NSGA-II et DLM-TLBO

6.5. Conclusion

Dans ce chapitre, un algorithme d'optimisation basé sur l'enseignement-apprentissage est proposé pour résoudre un problème discret, à grande échelle et multi-objectif connu dans le domaine de la robotique en essaim comme le problème de regroupement d'objets. Dans l'algorithme proposé, chaque individu sélectionne son processus d'apprentissage en fonction de son niveau défini selon son rang. Les meilleurs participent à une phase de collaboration, les apprenants moyens participent aux phases d'enseignement et d'apprentissage, tandis que les apprenants du dernier niveau participent uniquement à la phase d'enseignement. De plus, un taux d'apprentissage adaptatif est proposé pour définir l'effet d'un individu (apprenant) sur l'autre. Le taux d'apprentissage adaptatif définit les connaissances qui seront acquises en fonction du niveau. Les résultats montrent la supériorité de l'algorithme proposé par rapport à NSGA-II. Il a montré des performances prometteuses face au problème multi-objectif discret et à grande échelle présenté.

Conclusion générale

« Plus vous en savez, plus vous savez que vous ne savez pas. »

– Aristote

Les travaux de recherche menés dans cette thèse portent sur l'optimisation multi-objectif. La présente conclusion résume les propositions, les contributions de recherche, et les résultats obtenus et nos perspectives pour les futurs travaux de recherche. Souvent, les applications du monde réel ont plusieurs objectifs, généralement en conflit, à optimiser. Plusieurs méthodes de résolution ont été proposées pour ce type de problème. L'un des domaines d'application très intéressants est la robotique en essaim. Dans cette thèse, le problème de regroupement d'objets est choisi comme un cas d'étude.

Le problème de regroupement d'objets classique a comme objectif l'optimisation de la qualité du tas formé par les agents-robots. La première contribution de ce travail consiste à remodeler ce problème en un problème avec un autre objectif supplémentaire, qui est la consommation énergétique. Ce dernier objectif est ajouté puisque son optimisation devient de plus en plus importante dans un environnement d'agents-robots simples munis d'une capacité de batteries limitée et sans avoir la possibilité de rechargement de leur énergie, ce qui conduit à un avortement possible de la mission. L'algorithme évolutionnaire multi-objectif NSGA-II a été utilisé pour résoudre le problème de regroupement bi-objectif. Par la suite, un ensemble de solutions (modèles de comportements) a été présenté. Les résultats montrent que les bonnes solutions en termes de qualité de regroupement consomment plus d'énergie, mais qu'elles restent le meilleur compromis possible dans ce contexte. Aussi, l'existence d'une relation étroite entre le choix des comportements, la qualité du regroupement et l'énergie consommée a été montrée.

De l'autre part, l'une des meilleures solutions représentant un modèle comportemental bi-objectif est sélectionnée pour être évaluée dans différentes circonstances. La robustesse, la scalabilité et la flexibilité du système utilisant ce modèle comportemental ont été examinées. L'effet de quelques facteurs dynamiques et accidentels, qui peuvent influencer le fonctionnement du système, a été étudié. Aussi, deux modèles comportementaux bi-objectif sont comparés à un autre mono-objectif. Les résultats montrent la supériorité de l'approche de modélisation bi-objectif proposée. La solution bi-objectif est plus adaptée, car elle permet d'économiser l'énergie tout en offrant une meilleure qualité. De plus, elle est plus scalable et plus flexible qu'une solution à objectif unique.

Motiver par les caractéristiques de l'algorithme basé sur l'enseignement-apprentissage et afin d'améliorer la qualité du front de Pareto obtenu, une variante de ce dernier a été proposée pour résoudre le problème bi-objectif décrit ci-dessus. Dans l'algorithme proposé, chaque individu sélectionne son processus d'apprentissage en fonction de son rang. Les meilleurs individus participent à une phase de collaboration, les apprenants moyens participent aux phases d'enseignement et d'apprentissage, tandis que les apprenants du dernier niveau participent uniquement à la phase d'enseignement. Aussi, pour définir l'effet d'un individu (apprenant) sur l'autre et déterminer le flux de connaissances entre eux, un

taux d'apprentissage adaptatif est proposé. Les résultats montrent les performances prometteuses et la supériorité de l'algorithme proposé par rapport à NSGA-II.

Comme perspectives de cette thèse, nous envisageons de mener des travaux de recherche sur les points suivants :

(1) Utiliser un feedback négatif pour arrêter les agents-robots une fois la tâche est terminée. Dans les travaux déjà cités et le travail présenté dans cette thèse, le moment de terminaison de la tâche de regroupement d'objets est déterminé au niveau macro, en fixant un seuil de la qualité et/ou d'énergie consommée. Sinon les agents-robots poursuivent leur fonctionnement, ce qui pourrait mener à la déformation du tas d'objets. L'utilisation d'un feedback négatif déclenche un arrêt graduel au niveau micro.
(2) Aussi, le feedback négatif peut être utilisé pour minimiser l'énergie, en donnant par exemple un feedback négatif en cas d'existence de plusieurs agents-robots dans le même voisinage.
(3) Améliorer la diversité de l'algorithme proposé DLM-TLBO en utilisant le taux d'apprentissage pour équilibrer de manière auto-adaptative l'exploration et l'exploitation pendant le processus de recherche.
(4) Valider les performances de DLM-TLBO sur des problèmes d'optimisation multi-objectif discrets à grande échelle en utilisant une suite de problèmes de référence (benchmarks).

Bibliographie

[1] A. Messac, A. Ismail-Yahaya et C. A. Mattson, «The normalized normal constraint method for generating the Pareto frontier,» *Structural and multidisciplinary optimization,* vol. 25, n°2, pp. 86-98, 2003.

[2] R. V. Rao, V. J. Savsani et D. P. Vakharia, «Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems,» *Computer-Aided Design,* vol. 43, n°3, pp. 303-315, 2011.

[3] R. V. Rao, V. J. Savsani et D. P. Vakharia, «Teaching–learning-based optimization: an optimization method for continuous non-linear large scale problems,» *Information sciences,* vol. 183, n°1, pp. 1-15, 2012.

[4] T. D. Barfoot et G. M. D'eleuterio, «Evolving Distributed Control for an ObjectClustering Task,» chez *Stigmergy and Collective Robots, Artificial Life IV: Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Syst*, Cambridge, MA, 2005.

[5] M. Martin, B. Chopard et P. Albuquerque, «Formation of an ant cemetery: swarm intelligence or statistical accident?,» *Future Generation Computer Systems,* vol. 18, n°7, pp. 951-959, 2002.

[6] V. Hartmann, «Evolving agent swarms for clustering and sorting,» chez *Proceedings of the 7th Annual conference on Genetic and Evolutionary Computation*, Washington, USA., 2005.

[7] M. C. Cammaerts, «Some New Information on Ants' Cemeteries Organization,» *Asian journal of biology,* vol. 2, n°1, pp. 1-10, 2017.

[8] F. Zou, D. Chen et Q. Xu, «A Survey of Teaching–Learning-Based Optimization,» *Neurocomputing,* vol. 335, pp. 366-383, 2019.

[9] W. Aouadj, M.-R. Abdessemed et R. Seghir, «a Reliable Behavioral Model, Optimizing Energy Consumption and Object Clustering Quality by Naïve Robots,» *International journal of swarm intelligence research,* vol. 12, n°4, 2021.

[10] W. Aouadj, M.-R. Abdessemed et R. Seghir, «Discrete Large-scale Multi-Objective Teaching-Learning-Based Optimization Algorithm,» KENITRA, Morocco, 2021.

[11] R. Marler et J. Arora, «Survey of multi-objective optimization methods for engineering,» *Structural and multidisciplinary optimization,* vol. 26, n°6, pp. 369-395, April 2004.

[12] I. P. Stanimirovic, M. L. Zlatanovic et M. D. Petkovic, «On the linear weighted sum method for multi-objective optimization,» *Facta Acta Universitatis,* vol. 26, n°4, pp. 49-63, 2011.

[13] K. Ikeda et A. Hontani, «Probability of Perfect Reconstruction of Pareto Set in Multi-Objective Optimization,» chez *International Conference on Neural Information Processing*, 2013.

[14] K. Deb, Multi-Objective Optimization Using Evolutionary Algorithms, vol. 16, New York, NY, USA: John Wiley \& Sons, Inc, 2001.

[15] J. S. Arora, Introduction to optimum design, Elsevier, 2004.

[16] G. Chiandussi, M. Codegone, S. Ferrero et F. E. Varesio, «Comparison of multi-objective optimization methodologies for engineering applications,» *Computers & Mathematics with Applications,* vol. 63, n°5, pp. 912-942, 2012.

[17] S. Huband, P. Hingston, L. Barone et L. While, « A review of multiobjective test problems and a scalable test problem toolkit,» *IEEE Transactions on Evolutionary Computation,* vol. 10, n°5, pp. 477-506, 2006.

[18] V. Pareto, «Manuale di Economica Politica,» *Societa Editrice,* vol. 13, 1906.

[19] K. Deb et K. Miettinen, Multiobjective optimization: Interactive and evolutionary approaches, vol. 5252, Springer Science & Business Media, 2008.

[20] A. Chinchuluun et P. Pardalos, «A survey of recent developments in multiobjective optimization,» *Annals of Operations Research,* vol. 154, n°1, pp. 29-50, 2007.

[21] R. Marler et J. Arora, «The weighted sum method for multi-objective optimization: new insights,» *Structural and multidisciplinary optimization,* vol. 41, n°6, pp. 853-862, 2010.

[22] I. Y. Kim et O. L. De Weck, «Adaptive weighted sum method for multiobjective optimization: a new method for Pareto front generation,» *Structural and multidisciplinary optimization,* vol. 31, n°2, pp. 105-116, 2006.

[23] A. Charnes, W. Cooper et R. Ferguson, «Optimal estimation of executive compensation by linear programming,» *Management science,* vol. 1, n°2, pp. 138-151, 1955.

[24] Y. Haimes, L. Ladson et D. Wismer, «Bicriterion formulation of problems of integrated system identification and system optimization,» *IEEE Transactions on Systems Man and Cybernetics,* vol. 3, n°1971. , p. 296, 1971.

[25] Y. Fu et U. M. Diwekar, «An efficient sampling approach to multiobjective optimization,» *Annals of Operations Research,* vol. 132, n°1-4, pp. 109-134, 2004.

[26] I. P. Stanimirovic, «Compendious lexicographic method for multi-objective optimization,» *Ser. Math. Inform,* vol. 27, n°1, pp. 55-66, 2012.

[27] Y. Collette et P. Siarry, Optimisation multiobjectif: Algorithmes, Editions Eyrolles, 2011.

[28] C. M. Fonseca et P. J. Fleming, «Genetic Algorithms for Multiobjective Optimization: FormulationDiscussion and Generalization,» *Icga ,* vol. 93, pp. 416-423, 1993.

[29] D. Goldberg, «Genetic algorithms in search,» *Optimization and MachineLearning,* 1989.

[30] N. Srinivas et K. Deb, «Muiltiobjective optimization using nondominated sorting in genetic algorithms,» *Evolutionary computation,* vol. 2, n°3, pp. 221-248, 1994.

[31] I. Giagkiozis, R. C. Purshouse et P. J. Fleming, «An overview of population-based algorithms for multi-objective optimisation,» *International Journal of Systems Science,* vol. 46, n°9, pp. 1572-1599, 2015.

[32] K. Deb, A. Pratap, S. Agarwal et T. A. M. T. Meyarivan, «A fast and elitist multiobjective genetic algorithm: NSGA-II,» *IEEE transactions on evolutionary computation,* vol. 6, n°2, pp. 182-197, 2002.

[33] D. Beasley, D. R. Bull et R. R. Martin, «A sequential niche technique for multimodal function optimization,» *Evolutionary computation,* vol. 1, n°2, pp. 101-125, 1993.

[34] D. H. Loughlin et S. R. Ranjithan, «The Neighborhood Constraint Method: A Genetic Algorithm-Based Multiobjective Optimization Technique,» chez *ICGA*, 1997.

[35] D. E. Goldberg et J. Richardson, «Genetic algorithms with sharing for multimodal function optimization,» chez *Genetic algorithms and their applications: Proceedings of the Second International Conference on Genetic Algorithms*, NJ: Lawrence Erlbaum, 1987.

[36] C.Yann et S.Patrick, optimisation multiobjectif, Paris: Eyrolles, 2002.

[37] A. Przybylski et X. Gandibleux, «Multi-objective branch and bound,» *European Journal of Operational Research,* vol. 260, n°3, pp. 856-872, 2017.

[38] J. Mahmoudimehr et P. Sebghati, «A novel multi-objective Dynamic Programming optimization method: Performance management of a solar thermal power plant as a case study,» *Energy,* vol. 168, pp. 796-814, 2019.

[39] M. A. Ilgin et S. M. Gupta, «Physical programming: A review of the state of the art,» *Studies in Informatics and Control,* vol. 21, n°4, pp. 349-366, 2012.

[40] I. Das et J. Dennis, «Normal-Boundary Intersection: An Alternate Method for Generating Pareto Optimal Points in Multicriteria Optimization Problems,» NASA Langley Research Center, ICASE Report No. 96-62, Hampton, Virginia, 1996.

[41] I. Das, «On characterizing the “knee” of the Pareto curve based on normal-boundary intersection,» *Structural Optimization,* vol. 18, n°2-3, pp. 107-115, 1999.

[42] E. L. Ulungu, J. F. P. H. Teghem, P. H. Fortemps et D. Tuyttens, «MOSA method: a tool for solving multiobjective combinatorial optimization problems,» *Journal of multicriteria decision analysis,* vol. 8, n°4, p. 221, 1999.

[43] K. I. Smith, R. M. Everson, J. E. Fieldsend, C. Murphy et R. Misra, «Dominance-based multiobjective simulated annealing,» *IEEE Transactions on Evolutionary Computation,* vol. 12, n°3, pp. 323-342, 2008.

[44] D. M. Jaeggi, G. T. Parks, T. Kipouros et P. J. Clarkson, «The development of a multi-objective Tabu Search algorithm for continuous optimisation problems,» *European Journal of Operational Research,* vol. 185, n°3, pp. 1192-1212, 2008.

[45] J. E. C. Arroyo et A. A. de Souza Pereira, «A GRASP heuristic for the multi-objective permutation flowshop scheduling problem,» *The International Journal of Advanced Manufacturing Technology,* vol. 55, n°5-8, pp. 741-753, 2011.

[46] J. M. Lanza-Gutierrez et J. A. Gomez-Pulido, «Studying the multiobjective variable neighbourhood search algorithm when solving the relay node placement problem in wireless sensor networks,» *Soft Computing,* vol. 20, n°1, pp. 67-86, 2016.

[47] A. Alsheddy, «Empowerment scheduling: a multi-objective optimization approach using guided local search,» Doctoral dissertation, University of Essex, 2011.

[48] T. Dokeroglu, E. Sevinc, T. Kucukyilmaz et A. Cosar, «A survey on new generation metaheuristic algorithms,» *Computers & Industrial Engineering,* vol. 137, p. 106040, 2019.

[49] D. Simon, Evolutionary optimization algorithms, New Jersey: John Wiley & Sons, 2013.

[50] N. Nedjah et L. S. Junior, «Review of methodologies and tasks in swarm robotics towards standardization,» *Swarm and Evolutionary Computation,* vol. 100565, 2019.

[51] R. Datta et R. G. Regis, «A surrogate-assisted evolution strategy for constrained multi-objective optimization,» *Expert Systems with Applications,* vol. 57, pp. 270-284, 2016.

[52] B. Y. Qu et P. N. Suganthan, «Multi-objective evolutionary programming without non-domination sorting is up to twenty times faster,» chez *IEEE Congress on Evolutionary Computation* , Trondheim, Norway, 2009.

[53] H. Zhao, «A multi-objective genetic programming approach to developing Pareto optimal decision trees,» *Decision Support Systems,* vol. 43, n°3, pp. 809-826, 2007.

[54] F. Xue, A. C. Sanderson et R. J. Graves, «Pareto-based multi-objective differential evolution,» chez *Congress on Evolutionary Computation*, Canberra, ACT, Australia, 2003.

[55] C. C. Coello et M. S. Lechuga, «MOPSO: A proposal for multiple objective particle swarm optimization,» chez *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600)* , Honolulu, HI, USA , 2002.

[56] Y. Gao, H. Guan, Z. Qi, Y. Hou et L. Liu, «A multi-objective ant colony system algorithm for virtual machine placement in cloud computing,» *Journal of Computer and System Sciences,* vol. 79, n°8, p. 1230–1242, 2013.

[57] I. Aydin, M. Karakose et E. Akin, «A multi-objective artificial immune algorithm for parameter optimization in support vector machine,» *Applied soft computing,* vol. 11, n°1, pp. 120-129, 2011.

[58] B. Niu, H. Wang, J. Wang et L. Tan, «Multi-objective bacterial foraging optimization,» *Neurocomputing,* vol. 116, pp. 336-345, 2013.

[59] R. Akbari, R. Hedayatzadeh, K. Ziarati et B. Hassanizadeh, «A multi-objective artificial bee colony algorithm,» *Swarm and Evolutionary Computation,* vol. 2, pp. 39-52, 2012.

[60] K. Jamuna et K. S. Swarup, «Multi-objective biogeography based optimization for optimal PMU placement,» *Applied Soft Computing,* vol. 12, n°5, pp. 1503-1510, 2012.

[61] S. Mirjalili, «Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems,» *Neural Computing and Applications,* vol. 27, n°4, pp. 1053-1073, 2016.

[62] X. S. Yang et S. Deb, «Multiobjective cuckoo search for design optimization,» *Computers & Operations Research,* vol. 40, n°6, pp. 1616-1624, 2013.

[63] E. Emary, W. Yamany, A. E. Hassanien et V. Snasel, «Multi-objective gray-wolf optimization for attribute reduction,» *Procedia Computer Science,* vol. 65, pp. 623-632, 2015.

[64] I. Aljarah, M. Habib, H. Faris, N. Al-Madi, A. A. Heidari, M. Mafarja, M. Abd Elaziz et S. Mirjalili, «A dynamic locality multi-objective salp swarm algorithm for feature selection,» *Computers & Industrial Engineering,* vol. 147, p. 106628, 2020.

[65] H. R. Hassanzadeh et M. Rouhani, «A multi-objective gravitational search algorithm,» chez *2nd international conference on computational intelligence, communication systems and networks*, Liverpool, UK, 2010.

[66] A. Ranjbar, S. Talatahari et F. Hakimpour, «The application of multi-objective charged system search algorithm for optimization problems,» *Scientia Iranica,* vol. 26, n°3, pp. 1249-1265, 2019.

[67] F. Ebadifard et S. M. Babamir, «Optimizing multi objective based workflow scheduling in cloud computing using black hole algorithm,» chez *3th International Conference on Web Research (ICWR)*, Tehran, Iran, 2017.

[68] H. B. Tolabi, M. R. Shakarami, R. Hosseini et S. B. M. Ayob, «Novel FGbSA: Fuzzy-Galaxy-based search algorithm for multi-objective reconfiguration of distribution systems,» *Russian Electrical Engineering,* vol. 87, n°10, pp. 588-595, 2016.

[69] C. Dai et X. Lei, «A multiobjective brain storm optimization algorithm based on decomposition,» *Complexity,* vol. 2019, 2019.

[70] S. Sivasubramani et K. S. Swarup, «Multi-objective harmony search algorithm for optimal power flow problem,» *International Journal of Electrical Power & Energy Systems,* vol. 33, n°3, pp. 745-752, 2011.

[71] T. Niknam, R. Azizipanah-Abarghooee et M. R. Narimani, «A new multi objective optimization approach based on TLBO for location of automatic voltage regulators in distribution systems,» *Engineering Applications of Artificial Intelligence,* vol. 25, n°8, pp. 1577-1588, 2012.

[72] S. Akyol et B. Alatas, «Plant intelligence based metaheuristic optimization algorithms,» *Artif Intell Rev ,* vol. 47, p. 417–462, 2017.

[73] X. S. Yang, M. Karamanoglu et X. He, «Multi-objective flower algorithm for optimization,» *Procedia Computer Science,* vol. 18, pp. 861-868, 2013.

[74] A. H. Halim et I. Ismail, «Nonlinear plant modeling using neuro-fuzzy system with Tree Physiology Optimization,» chez *IEEE Student Conference on Research and Developement*, Putrajaya, Malaysia, 2013.

[75] A. Goli, E. B. Tirkolaee, B. Malmir, G. B. Bian et A. K. Sangaiah, «A multi-objective invasive weed optimization algorithm for robust aggregate production planning under uncertain seasonal demand,» *Computing,* vol. 10, n°6, pp. 499-529, 2019.

[76] A. Mukhopadhyay, U. Maulik, S. Bandyopadhyay et C. A. C. Coello, «A survey of multiobjective evolutionary algorithms for data mining: Part I.,» *IEEE Transactions on Evolutionary Computation,* vol. 18, n°1, pp. 4-19, 2014.

[77] A. Mukhopadhyay, U. Maulik, S. Bandyopadhyay et C. A. C. Coello, «Survey of multiobjective evolutionary algorithms for data mining: part II,» *IEEE Transactions on Evolutionary Computation,* vol. 18, n°1, pp. 20-35, 2014.

[78] R. Farmani, D. A. Savic et G. A. Walters, «Evolutionary multi-objective optimization in water distribution network design,» *Engineering Optimization,* vol. 37, n°2, pp. 167-183, 2005.

[79] P. M. Reed, D. Hadka, J. D. Herman, J. R. Kasprzyk et J. B. Kollat, «Evolutionary multiobjective optimization in water resources: The past, present, and future,» *Advances in water resources,* vol. 51, pp. 438-456, 2013.

[80] L. S. D. Oliveira et S. F. Saramago, «Multiobjective optimization techniques applied to engineering problems,» *Journal of the brazilian society of mechanical sciences and engineering,* vol. 32, n°1, pp. 94-105, 2010.

[81] O. B. Augusto, F. Bennis et S. Caro, «Multiobjective engineering design optimization problems: a sensitivity analysis approach,» *Pesquisa Operacional,* vol. 32, n°3, pp. 575-596, 2012.

[82] Y. Donoso et R. Fabregat, Multi-objective optimization in computer networks using metaheuristics, CRC Press., 2016.

[83] M. Gamal, E. Morsy et A. Fathy, «Multi-Objective transmitters Placement Problem in wireless Networks,» chez *Proceedings of the Sixth International Symposium on Information and Communication Technology*, 2015, December.

[84] Z. Fei, B. Li, C. Xing, S. Yang, H. Chen et L. Hanzo, «Multi-objective optimization in wireless sensor networks,» *IEEE Communications Surveys & Tutorials,* 2016.

[85] A. Peiravi, H. R. Mashhadi et S. Hamed Javadi, «An optimal energy-efficient clustering method in wireless sensor networks using multi-objective genetic algorithm,» *International Journal of Communication Systems,* vol. 26, n°11, pp. 114-126, 2013.

[86] S. M. Jameii, K. Faez et M. Dehghan, «Multi-objective energy efficient optimization algorithm for coverage control in wireless sensor networks,» *Int. J. Comput. Sci. Eng. Inf. Technol,* vol. 3, pp. 25-33, 2013.

[87] Y. Gao, H. Guan, Z. Qi, Y. Hou et L. Liu, «A multi-objective ant colony system algorithm for virtual machine placement in cloud computing,» *Journal of Computer and System Sciences,* vol. 79, n°8, pp. 1230-1242, 2013.

[88] M. Mezmaz, N. Melab, Y. Kessaci, Y. C. Lee, E. G. Talbi, A. Y. Zomaya et D. Tuyttens, «A parallel bi-objective hybrid metaheuristic for energy-aware scheduling for cloud computing systems,» *Journal of Parallel and Distributed Computing,* vol. 71, n°11, pp. 1497-1508, 2011.

[89] M. Fadaee et M. A. M. Radzi, «Multi-objective optimization of a stand-alone hybrid renewable energy system by using evolutionary algorithms: A review,» *Renewable and Sustainable Energy Reviews,* vol. 16, n°5, pp. 3364-3369, 2012.

[90] C. A. Nicolaou et N. Brown, «Multi-objective optimization methods in drug design,» *Drug Discovery Today: Technologies,* vol. 10, n°3, pp. e427-e435, 2013.

[91] J. García, R. Iglesias, M. A. Rodríguez et C. V. Regueiro, «Incremental reinforcement learning for multi-objective robotic tasks,» *Knowledge and Information Systems,* pp. 1-30, 2016.

[92] Y. Zhang, D. W. Gong et J. H. Zhang, «Robot path planning in uncertain environment using multi-objective particle swarm optimization,» *Neurocomputing,* vol. 103, pp. 172-185, 2013.

[93] J. A. Herrera Ortiz, K. Rodríguez-Vázquez, M. A. Padilla Castañeda et F. Arámbula Cosío, « Autonomous robot navigation based on the evolutionary multi-objective optimization of potential fields,» *Engineering Optimization,* vol. 45, n°1, pp. 19-43, 2013.

[94] A. Arias-Montano, C. A. C. Coello et E. Mezura-Montes, «Multiobjective evolutionary algorithms in aeronautical and aerospace engineering,» *IEEE Transactions on Evolutionary Computation,* vol. 16, n°5, pp. 662-694, 2012.

[95] M. E. Beniakar, A. G. Sarigiannidis, P. E. Kakosimos et A. G. Kladas, «Multiobjective evolutionary optimization of a surface mounted PM actuator with fractional slot winding for aerospace applications,» *IEEE transactions on magnetics,* vol. 50, n°2, pp. 665-668, 2014.

[96] Y. Wang, H. Yin, S. Zhang et X. Yu, «Multi-objective optimization of aircraft design for emission and cost reductions,» *Chinese Journal of Aeronautics,* vol. 27, n°1, pp. 52-53, 2014.

[97] W. Stadler, Multicriteria Optimization in Engineering and in the Sciences, Springer Science & Business Media, 2013.

[98] D. Yu, J. Hong, J. Zhang et Q. Niu, «Multi-Objective Individualized-Instruction Teaching-Learning-Based Optimization Algorithm,» *Applied Soft Computing,* vol. 62, n°1, pp. 288-314, 2018.

[99] H. C. Tsai, «Confined teaching-learning-based optimization with variable search strategies for continuous optimization,» *Information Sciences,* vol. 500, pp. 34-47, 2019.

[100] M. Kaboli et M. Akhlaghi, «Binary teaching-learning-based optimization algorithm is used to investigate the superscattering plasmonic nanodisk,» *Optics and Spectroscopy,* vol. 120, n°6, pp. 958-963, 2016.

[101] M. Allam et M. Nandhini, «Optimal feature selection using binary teaching learning based optimization algorithm,» *Journal of King Saud University-Computer and Information Sciences,* 2018.

[102] D. Chen, F. Zou, R. Lu, L. Yu, Z. Li et J. Wang, «Multi-objective optimization of community detection using discrete teaching–learning-based optimization with decomposition,» *Information Sciences,* vol. 369, n°1, pp. 402-418, 2016.

[103] S. Talatahari et V. Goodarzimehr, «A Discrete Hybrid Teaching-Learning-Based Optimization algorithm for optimization of space trusses,» *Journal of Structural Engineering and Geo-Techniques,* vol. 9, n°1, 2019.

[104] E. Natarajan, V. Kaviarasan, W. H. Lim, S. S. Tiang, S. Parasuraman et S. Elango, «Non-dominated sorting modified teaching–learning-based optimization for multi-objective machining of polytetrafluoroethylene (PTFE),» *Journal of Intelligent Manufacturing,* pp. 1-25, 2019.

[105] V. K. Patel et V. J. Savsani, «A multi-objective improved teaching–learning based optimization algorithm (MO-ITLBO),» *Information Sciences,* vol. 357, n°1, pp. 182-200, 2016.

[106] R. Xue et Z. Wu, «A survey of application and classification on teaching-learning-based optimization algorithm,» *IEEE Access,* vol. 8, pp. 1062-1079, 2019.

[107] W. Shao, D. Pi et Z. Shao, «An extended teaching-learning based optimization algorithm for solving no-wait flow shop scheduling problem,» *Applied Soft Computing,* vol. 61, pp. 193-210, 2017.

[108] P. K. Roy et R. Sarkar, «Solution of unit commitment problem using quasi-oppositional teaching learning based algorithm,» *International Journal of Electrical Power & Energy Systems,* vol. 60, n°11, pp. 96-106, 2014.

[109] S. Sultana et P. K. Roy, «Multi-objective quasi-oppositional teaching learning based optimization for optimal location of distributed generator in radial distribution systems,» *International Journal of Electrical Power & Energy Systems,* vol. 63, n°12, pp. 534-545, 2014.

[110] J. Cao et J. Luo, «A study on SVM based on the weighted elitist teaching-learning-based optimization and application in the fault diagnosis of chemical process,» chez *MATEC web of conferences*, 2015.

[111] S. C. Satapathy, A. Naik et K. Parvathi, «Weighted teaching-learning-based optimization for global function optimization,» *Applied Mathematics,* vol. 4, n°3, pp. 429-439, 2013.

[112] Z. S. Wu, W. P. Fu et R. Xue, «Nonlinear inertia weighted teaching-learning-based optimization for solving global optimization problem,» *Computational intelligence and neuroscience,* vol. 2015, n°87, 2015.

[113] G. Li, P. Niu, W. Zhang et Y. Liu, «Model NOx emissions by least squares support vector machine with tuning based on ameliorated teaching–learning-based optimization,» *Chemometrics and Intelligent Laboratory Systems,* vol. 126, n°8, pp. 11-20, 2013.

[114] S. M. A. Bulbul et P. K. Roy, «Adaptive teaching learning based optimization applied to nonlinear economic load dispatch problem,» *International Journal of Swarm Intelligence Research (IJSIR),* vol. 5, n°4, pp. 1-16, 2014.

[115] D. Chen, R. Lu, F. Zou et S. Li, «Teaching-learning-based optimization with variable-population scheme and its application for ANN and global optimization,» *Neurocomputing,* vol. 173, pp. 1096-1111, 2016.

[116] S. Sleesongsom et S. Bureerat, «Four-bar linkage path generation through self-adaptive population size teaching-learning based optimization,» *Knowledge-Based Systems,* vol. 135, pp. 180-191, 2017.

[117] R. Rao et V. Patel, «An elitist teaching-learning-based optimization algorithm for solving complex constrained optimization problems,» *International Journal of Industrial Engineering Computations,* vol. 3, n°4, pp. 535-560, 2012.

[118] B. Dong, X. Wu et Y. Sun, «A collaborative learning model inteaching-learning-based optimization: Some numerical results,» chez *Bio-inspired Computing—Theories and Applications*, Singapore, 2017.

[119] F. Ge, L. Hong et L. Shi, « An autonomous teaching-learning based optimization algorithm for single objective global optimization,» *International Journal of Computational Intelligence Systems,* vol. 9, n°3, pp. 506-524, 2016.

[120] X. Ji, H. Ye, J. Zhou, Y. Yin et X. Shen, «An improved teaching-learning-based optimization algorithm and its application to a combinatorial optimization problem in foundry industry,» *Applied Soft Computing,* vol. 57, pp. 504-516, 2017.

[121] A. Farah, T. Guesmi, H. H. Abdallah et A. Ouali, «A novel chaotic teaching–learning-based optimization algorithm for multi-machine power system stabilizers design problem,» *International Journal of Electrical Power & Energy Systems,* vol. 77, pp. 197-209, 2016.

[122] P. R. Krishna et S. Sao, «An improved TLBO algorithm to solve profit based unit commitment problem under deregulated environment,» chez *Procedia Technology*, Kottayam, Kerala, India, 2016.

[123] R. V. Rao et V. Patel, «An improved teaching-learning-based optimization algorithm for solving unconstrained optimization problems,» *Scientia Iranica,* vol. 20, n°3, pp. 710-720, 2013.

[124] B. D. Raja, R. L. Jhala et V. Patel, «Multi-objective optimization of a rotary regenerator using tutorial training and self-learning inspired teaching-learning based optimization algorithm (TS-TLBO),» *Applied Thermal Engineering,* vol. 93, pp. 456-467, 2016.

[125] F. Zou, L. Wang, X. Hei et D. Chen, «Teaching–learning-based optimization with learning experience of other learners and its application,» *Applied Soft Computing,* vol. 37, pp. 725-736, 2015.

[126] K. Yu, X. Wang et Z. Wang, «Constrained optimization based on improved teaching–learning-based optimization algorithm,» *Information Sciences,* vol. 352, pp. 61-78, 2016.

[127] F. Zou, L. Wang, X. Hei, D. Chen et D. Yang, «Teaching–learning-based optimization with dynamic group strategy for global optimization,» *Information sciences,* vol. 273, pp. 112-131, 2014.

[128] S. Reddy, «Clustered adaptive teaching–learning-based optimization algorithm for solving the optimal generation scheduling problem,» *Electrical Engineering,* vol. 100, n°1, pp. 333-346, 2018.

[129] F. Zou, D. Chen, R. Lu et P. Wang, «Hierarchical multi-swarm cooperative teaching–learning-based optimization for global optimization,» *Soft Computing,* vol. 21, n°23, pp. 6983-7004, 2017.

[130] D. L. Gonzalez-Alvarez, M. A. Vega-Rodriguez et A. Rubio-Largo, «Finding patterns in protein sequences by using a hybrid multiobjective teaching learning based optimization algorithm,» *IEEE/ACM transactions on computational biology and bioinformatics,* vol. 12, n°3, pp. 656-666, 2014.

[131] D. Chen, F. Zou, J. Wang et W. Yuan, «SAMCCTLBO: a multi-class cooperative teaching–learning-based optimization algorithm with simulated annealing,» *Soft Computing,* vol. 20, n°5, pp. 1921-1943, 2016.

[132] X. Qu, B. Liu, Z. Li, W. Duan, R. Zhang, W. Zhang et H. Li, «A novel improved teaching-learning based optimization for functional optimization,» chez *IEEE International Conference on Control and Automation* , 2016.

[133] A. Liu, X. Deng, Z. Tong, Y. Luo et B. Liu, «A simultaneous perturbation stochastic approximation enhanced teaching-learning based optimization,» chez *IEEE Congress on Evolutionary Computation (CEC)* , 2016.

[134] Z. Wang, R. Lu, D. Chen et F. Zou, «An experience information teaching–learning-based optimization for global optimization,» *IEEE Transactions on Systems, Man, and Cybernetics: Systems,* vol. 46, n°9, pp. 1202-1214, 2016.

[135] L. Wang, F. Zou, X. Hei, D. Yang, D. Chen et Q. Jiang, «An improved teaching–learning-based optimization with neighborhood search for applications of ANN,» *Neurocomputing,* vol. 143, n°16, pp. 231-247, 2014.

[136] M. Ghasemi, M. Taghizadeh et S. Ghavidel, «Solving optimal reactive power dispatch problem using a novel teaching–learning-based optimization algorithm,» *Engineering Applications of Artificial Intelligence,* vol. 39, pp. 100-108, 2015.

[137] T. Dokeroglu, «Hybrid teaching–learning-based optimization algorithms for the quadratic assignment problem,» *Computers & Industrial Engineering,* vol. 85, pp. 86-101, 2015.

[138] Z. Zhai, G. Jia et K. Wang, «A novel teaching-learning-based optimization with error correction and cauchy distribution for path planning of unmanned air vehicle,» *Computational intelligence and neuroscience,* vol. 3, pp. 1-12, 2018.

[139] M. Hamzeh, B. Vahidi et A. F. Nematollahi, «Optimizing configuration of cyber network considering graph theory structure and teaching–learning-based optimization (GT-TLBO),» *IEEE Transactions on Industrial Informatics,* vol. 15, n°4, pp. 2083-2090, 2018.

[140] T. Cheng, M. Chen, P. J. Fleming, Z. Yang et S. Gan, «A novel hybrid teaching learning based multi-objective particle swarm optimization,» *Neurocomputing,* vol. 222, n°1, pp. 11-25, 2017.

[141] S. Shahbeig, M. S. Helfroush et A. Rahideh, «A fuzzy multi-objective hybrid TLBO–PSO approach to select the associated genes with breast cancer,» *Signal processing,* vol. 131, pp. 58-65, 2017.

[142] F. Dib et I. Boumhidi, «Hybrid algorithm DE–TLBO for optimal H∞ and PID control for multi-machine power system,» *International Journal of System Assurance Engineering and Management,* vol. 8, n°2, pp. 925-936, 2017.

[143] M. Güçyetmez et E. Çam, «A new hybrid algorithm with genetic-teaching learning optimization (G-TLBO) technique for optimizing of power flow in wind-thermal power systems,» *Electrical engineering,* vol. 98, n°2, pp. 145-157, 2016.

[144] J. Huang, L. Gao et X. Li, «An effective teaching-learning-based cuckoo search algorithm for parameter optimization problems in structure designing and machining processes,» *Applied Soft Computing,* vol. 36, pp. 349-356, 2015.

[145] M. Zhang, Y. Pan, J. Zhu et G. Chen, «ABC-TLBO: A hybrid algorithm based on artificial bee colony and teaching-learning-based optimization,» chez *37th Chinese Control Conference (CCC)*, Wuhan, China, 2018.

[146] H. Ouyang, G. Ma, G. Liu, Z. Li et X. Zhong, «Hybrid teaching-learning based optimization with harmony search for engineering optimization problems,» chez *36th Chinese Control Conference (CCC)* , 2017.

[147] J. Nayak, B. Naik, H. S. Behera et A. Abraham, «Elitist teaching–learning-based optimization (ETLBO) with higher-order Jordan Pi-sigma neural network: a comparative performance analysis,» *Neural Computing and Applications,* vol. 30, n°5, pp. 1445-1468, 2018.

[148] B. Hemalatha et N. Rajkumar, «A versatile approach for dental age estimation using fuzzy neural network with teaching learning-based optimization classification,» *Multimedia Tools and Applications,* vol. 79, n°5, pp. 3645-3665, 2020.

[149] M. M. Puralachetty, V. K. Pamula, L. M. Gondela et V. N. B. Akula, «Teaching-learning-based optimization with two-stage initialization,» chez *IEEE Students' Conference on Electrical, Electronics and Computer Science (SCEECS)*, 2016.

[150] F. Zou, L. Wang, X. Hei, D. Chen et B. Wang, «Multi-objective optimization using teaching-learning-based optimization algorithm,» *Engineering Applications of Artificial Intelligence,* vol. 26, n°4, pp. 1291-1300, 2013.

[151] W. Lin, D. Y. Yu, S. Wang, C. Zhang, S. Zhang, H. Tian et S. ... Liu, «Multi-objective teaching–learning-based optimization algorithm for reducing carbon emissions and operation time in turning operations,» *Engineering Optimization,* vol. 47, n°7, pp. 994-1007, 2015.

[152] L. Li et J. Zheng, «Muoti Objective Genetic Algorithm Baesd on Pareto Front,» *Natural Science Jouranl- Xiangtan University ,* vol. 26, n°1, pp. 39-41, 2004.

[153] R. Rao et V. Patel, «Multi-objective optimization of two stage thermoelectric cooler using a modified teaching–learning-based optimization algorithm,» *Engineering Applications of Artificial Intelligence,* vol. 26, n°1, pp. 430-445, 2013a.

[154] R. V. Rao et V. Patel, «Multi-objective optimization of heat exchangers using a modified teaching-learning-based optimization algorithm,» *Applied Mathematical Modelling,* vol. 37, n°3, pp. 1147-1162, 2013b.

[155] K. Yu, X. Wang et Z. Wang, «Self-adaptive multi-objective teaching-learning-based optimization and its application in ethylene cracking furnace operation optimization,» *Chemometrics and Intelligent Laboratory Systems,* vol. 146, n°1, pp. 198-210, 2015.

[156] K. Yu, L. While, M. Reynolds, X. Wang, J. J. Liang, L. Zhao et Z. Wang, «Multiobjective optimization of ethylene cracking furnace system using self-adaptive multiobjective teaching-learning-based optimization,» *Energy,* vol. 148, n°1, pp. 469-481, 2018.

[157] S. Sahu, A. K. Barisal et A. Kaudi, «Multi-objective optimal power flow with DG placement using TLBO and MIPSO: A comparative study,» *Energy Procedia,* vol. 117, pp. 236-243, 2017.

[158] L. Xiao, Q. Zhu, C. Li, Y. Cao, Y. Tan et L. Li, «Application of modified teaching-learning algorithm in coordination optimization of TCSC and SVC,» chez *Chinese Conference on Pattern Recognition*, Changsha, China, 2014.

[159] A. Fathy, I. Ziedan et D. Amer, «Improved teaching–learning-based optimization algorithm-based maximum power point trackers for photovoltaic system,» *Electrical Engineering,* vol. 100, n°3, pp. 1773-1784, 2018.

[160] E. D. Collins et B. Ramachandran, « Power management in a microgrid using teaching learning based optimization algorithm,» chez *SoutheastCon 2017*, North Carolina,USA, 2017.

[161] B. I. Xiaojun et P. A. N. Tiewen, «An image enhancement method based on improved teaching-learning-based optimization algorithm,» *Journal of Harbin Engineering University,* vol. 37, n°12, pp. 1716-1721, 2016.

[162] V. S. Thakur, S. Gupta et K. Thakur, «Optimal quantization table generation for efficient satellite image compression using teaching learning based optimization technique,» chez *International Conference on Big Data Analytics and Computational Intelligence*, 2017.

[163] M. Kumar et S. K. Mishra, «Teaching learning based optimization-functional link artificial neural network filter for mixed noise reduction from magnetic resonance image,» *Bio-medical materials and engineering,* vol. 28, n°6, pp. 643-654, 2017.

[164] A. Aouf, L. Boussaid et A. Sakly, «TLBO-based adaptive neurofuzzy controller for mobile robot navigation in a strange environment,» *Computational intelligence and neuroscience,* vol. 2018, 2018.

[165] R. V. Rao, «Design optimization of a robot manipulator using TLBO and ETLBO algorithms,» chez *Teaching Learning Based Optimization Algorithm*, Cham, Springer, 2016, pp. 163-169.

[166] G. Sharma et A. Kumar, «Modified energy-efficient range-free localization using teaching–learning-based optimization for wireless sensor networks,» *IETE Journal of Research,* vol. 64, n°1, pp. 124-138, 2018.

[167] S. K. Panigrahi et S. Pattnaik, «Empirical study on clustering based on modified teaching learning based optimization,» *Procedia Computer Science,* vol. 92, pp. 442-449, 2016.

[168] S. Mohanty et S. Padhy, « A novel OFS–TLBO–SVR hybrid model for optimal budget allocation of government schemes to maximize GVA at factor cost,» *Journal of Management Analytics,* vol. 5, n°1, pp. 32-53, 2018.

[169] M. Shahrouzi, M. Aghabaglou et F. Rafiee, «Observer-teacher-learner-based optimization: An enhanced meta-heuristic for structural sizing design,» *Structural engineering and mechanics: An international journal,* vol. 62, n°5, pp. 537-550, 2017.

[170] R. Buddala et S. S. Mahapatra, «Two-stage teaching-learning-based optimization method for flexible job-shop scheduling under machine breakdown,» *The International Journal of Advanced Manufacturing Technology,* vol. 100, n°5, pp. 1419-1432, 2019.

[171] V. Trianni, Evolutionary swarm robotics: evolving self-organising behaviours in groups of autonomous robots, vol. 108, Berlin Heidelberg: Springer, 2008.

[172] V. Trianni, E. Tuci, C. Ampatzis et M. Dorigo, «Evolutionary swarm robotics: A theoretical and methodological itinerary from individual neuro-controllers to collective behaviours,» *The horizons of evolutionary robotics,* vol. 153, 2014.

[173] G. Seeja et A. Selvakumar, «A Survey on Swarm Robotic Modeling, Analysis and Hardware Architecture,» *Procedia Computer Science,* vol. 133, pp. 478-485, 2018.

[174] J. H. Lee et C. W. Ahn, «Improving energy efficiency in cooperative foraging swarm robots using behavioral model,» chez *Sixth International Conference on Bio-Inspired Computing: Theories and Applications*, Penang, Malaysia, 2011.

[175] I. Navarro et F. Matía, «An introduction to swarm robotics,» *ISRN robotics,* pp. 1-10, 2013.

[176] E. Şahin, «Swarm robotics: From sources of inspiration to domains of application,» chez *International workshop on swarm robotics*, Berlin, Heidelberg, 2004.

[177] H. Hamann, Swarm robotics: A formal approach, Springer International Publishing, 2018.

[178] M. Dorigo et T. Stützle, «Ant colony optimization: overview and recent advances,» chez *Handbook of metaheuristics*, Cham., Springer, 2019, pp. 311-351.

[179] S. Mirjalili, P. Jangir et S. Saremi, «Multi-objective ant lion optimizer: a multi-objective optimization algorithm for solving engineering problems,» *Applied Intelligence,* vol. 46, n°1, pp. 79-95, 2017.

[180] B. Yuce, E. Mastrocinque, M. S. Packianather, A. Lambiase et D. T. Pham, «The Bees Algorithm and its applications,» chez *Handbook of research on artificial intelligence techniques and algorithms*, IGI Global, 2015, pp. 122-151.

[181] M. Neshat, G. Sepidnam, M. Sargolzaei et A. N. Toosi, «Artificial fish swarm algorithm: a survey of the state-of-the-art, hybridization, combinatorial and indicative applications,» *Artificial intelligence review,* vol. 42, n°4, pp. 965-997, 2014.

[182] D. S. Cambuí et A. Rosas, «Density induced transition in a school of fish,» *Physica A: Statistical Mechanics and its Applications,* vol. 391, n°15, pp. 3908-3914, 2012.

[183] D. B. Kearns, «A field guide to bacterial swarming motility,» *Nature Reviews Microbiology,* vol. 8, n°9, pp. 634-644, 2010.

[184] W. Kim, T. Killam, V. Sood et M. G. Surette, «Swarm-cell differentiation in Salmonella enterica serovar Typhimurium results in elevated resistance to multiple antibiotics,» *Journal of bacteriology,* vol. 185, n°10, pp. 3111-3117, 2003.

[185] G. F. Young, L. Scardovi, A. Cavagna, I. Giardina et N. E. Leonard, «Starling flock networks manage uncertainty in consensus at low cost,» *PLoS computational biology,* vol. 9, n°1, p. e1002894, 2013.

[186] O. Peleg, J. M. Peters, M. K. Salcedo et L. Mahadevan, «Collective mechanical adaptation of honeybee swarms,» *Nature Physics,* vol. 14, n°12, pp. 1193-1198, 2018.

[187] A. Ward et M. Webster, «Collective Decision-Making,» chez *Sociality: The Behaviour of Group-Living Animals*, Cham., Springer, 2016, pp. 149-174.

[188] C. R. Reid, M. J. Lutz, S. Powell, A. B. Kao, I. D. Couzin et S. Garnier, «Army ants dynamically adjust living bridges in response to a cost–benefit trade-off,» chez *Proceedings of the National Academy of Sciences*, 2015.

[189] J. M. Graham, A. B. Kao, D. A. Wilhelm et S. Garnier, «Optimal construction of army ant living bridges,» *Journal of theoretical biology,* vol. 435, n°1, pp. 184-198, 2017.

[190] G. Beni, «From swarm intelligence to swarm robotics,» chez *International Workshop on Swarm Robotics* , Berlin, Heidelberg, 2004.

[191] L. Bayındır, «A review of swarm robotics tasks,» *Neurocomputing,* vol. 172, n° %11, pp. 292-321, 2016.

[192] Y. Tan et Z. Y. Zheng, «Research advance in swarm robotics,» *Defence Technology,* vol. 9, n°1, pp. 18-39, 2013.

[193] J. Thangavelautham, T. D. Barfoot et G. M. D'Eleuterio, «Evolutionary-Based Control Approaches for Multirobot Systems,» chez *Frontiers in Evolutionary Robotics*, Vienna, Austria, IntechOpen, 2008, pp. 36-60.

[194] K. Lerman et A. Galstyan, «Mathematical model of foraging in a group of robots:Effect of interference,» *Autonomous Robots,* vol. 13, p. 127–141, 2002.

[195] J. H. Holland, Emergence - From chaos to order, New York: Oxford University Press, 1998.

[196] G. Vorobyev, A. Vardy et W. Banzhaf, «Supervised learning in robotic swarms: From training samples to emergent behavior,» chez *Distributed Autonomous Robotic Systems*, Berlin, Heidelberg, 2014.

[197] A. Dussutour, V. Fourcassié, D. Helbing et J.-L. Deneubourg, «Optimal traffic organization in ants under crowded conditions,» *Nature,* vol. 428, pp. 70-73, 2004.

[198] T. J. Czaczkes, C. Grüter et F. L. Ratnieks, «Negative feedback in ants: crowding results in less trail pheromone deposition,» *Journal of the Royal Society Interface,* vol. 10, n°81, p. 20121009, 2013.

[199] E. Bonabeau, D. D. R. D. F. Marco, M. Dorigo et G. Theraulaz, Swarm intelligence: from natural to artificial systems, New York, USA: Oxford university press, 1999.

[200] J. D. Bjerknes et A. F. Winfield, «On fault tolerance and scalability of swarm robotic systems,» chez *Distributed autonomous robotic systems*, Berlin, Heidelberg, 2013.

[201] M. Dorigo, M. Birattari et M. Brambilla, «Swarm robotics,» *Scholarpedia,* vol. 9, n°1, p. 1463, 2014.

[202] M. Al Haek, A. Ismail, A. Basalib et N. Makarim, «Exploring energy charging problem in swarm robotic systems using foraging simulation,» *Jurnal Teknologi,* vol. 76, n°1, pp. 239-244, 2015.

[203] L. Bayindir et E. Şahin, «A review of studies in swarm robotics,» *Turkish Journal of Electrical Engineering & Computer Sciences,* vol. 15, n°2, pp. 115-147, 2007.

[204] M. Brambilla, E. Ferrante, M. Birattari et M. Dorigo, «Swarm robotics: a review from the swarm engineering perspective,» *Swarm Intelligence,* vol. 7, n°1, pp. 1-41, 2013.

[205] Y. U. Cao, A. S. Fukunaga et A. Kahng, «Cooperative mobile robotics: Antecedents and directions,» *Autonomous robots,* vol. 4, n°1, pp. 7-27, 1997.

[206] L. Iocchi, D. Nardi et M. Salerno, «Reactivity and deliberation: a survey on multi-robot systems,» chez *Workshop on Balancing Reactivity and Social Deliberation in Multi-Agent Systems*, 2000.

[207] M. Brambilla, C. Pinciroli, M. Birattari et M. Dorigo, «Property-driven design for swarm robotics,» chez *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*, Valencia, Spain, 2012.

[208] W. Li, M. Gauci et R. Groß, «Turing learning: a metric-free approach to inferring behavior and its application to swarms,» *Swarm Intelligence,* vol. 10, n°3, pp. 211-243, 2016.

[209] F. Ducatelle, A. Förster, G. A. Di Caro et L. M. Gambardella, «New task allocation methods for robotic swarms,» chez *9th IEEE/RAS conference on autonomous robot systems and competitions*, Castelo Branco, Portugal, 2009.

[210] A. L. Christensen, R. OGrady et M. Dorigo, «From fireflies to fault-tolerant swarms of robots,» *IEEE Transactions on Evolutionary Computation,* vol. 13, n°4, pp. 754-766, 2009.

[211] M. Gauci, J. Chen, T. J. Dodd et R. Groß, «Evolving aggregation behaviors in multi-robot systems with binary sensors,» chez *Distributed autonomous robotic systems*, Berlin, Heidelberg, 2014.

[212] R. Jeanson, C. Rivault, J. L. Deneubourg, S. Blanco, R. Fournier, C. Jost et G. Theraulaz, «Self-organized aggregation in cockroaches,» *Animal behaviour,* vol. 69, n°1, pp. 169-180, 2005.

[213] F. Arvin, K. Samsudin, A. R. Ramli et M. Bekravi, «Imitation of honeybee aggregation with collective behavior of swarm robots,» *International Journal of Computational Intelligence Systems,* vol. 4, n°4, pp. 739-748, 2011.

[214] F. Arvin, A. E. Turgut et S. Yue, «Fuzzy-based aggregation with a mobile robot swarm,» chez *International Conference on Swarm Intelligence*, Shenzhen, China, 2012.

[215] A. M. Halász, Y. Liang, M. A. Hsieh et H. J. Lai, «Emergence of specialization in a swarm of robots,» chez *Distributed Autonomous Robotic Systems*, Berlin, Springer, 2013, pp. 403-416.

[216] A. Brutschy, G. Pini, C. Pinciroli, M. Birattari et M. Dorigo, «Self-organized task allocation to sequentially interdependent tasks in swarm robotics,» *Autonomous agents and multi-agent systems,* vol. 28, n°1, pp. 101-125, 2014.

[217] G. Valentini, H. Hamann et M. Dorigo, «Global-to-local design for self-organized task allocation in swarms,» Bruxelles, Belgium, 2016.

[218] H. Lau, J. Timmis et I. Bate, «Collective self-detection scheme for adaptive error detection in a foraging swarm of robots,» chez *International Conference on Artificial Immune Systems*, Cambridge, UK, 2011.

[219] A. G. Millard, J. Timmis et A. F. Winfield, «Towards exogenous fault detection in swarm robotic systems,» chez *Conference towards Autonomous Robotic Systems*, Oxford, UK, 2013.

[220] F. Ducatelle, G. A. Di Caro, C. Pinciroli, F. Mondada et L. Gambardella, «Communication assisted navigation in robotic swarms: self-organization and cooperation,» chez *IEEE/RSJ International Conference on Intelligent Robots and Systems*, San Francisco, CA, USA, 2011.

[221] R. Mayet, J. Roberz, T. Schmickl et K. Crailsheim, «Antbots: A feasible visual emulation of pheromone trails for swarm robots,» chez *International Conference on Swarm Intelligence*, Brussels, Belgium, 2010.

[222] G. Vásárhelyi, C. Virágh, G. Somorjai, N. Tarcai, T. Szörényi, T. Nepusz et T. Vicsek, «Outdoor flocking and formation flight with autonomous aerial robots,» chez *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Chicago, IL, USA, 2014.

[223] M. Carrillo, I. Gallardo, J. Del Ser, E. Osaba, J. Sanchez-Cubillo, M. N. Bilbao, A. G´alvez et A. Iglesias, «A Bio-inspired Approach for Collaborative Exploration with Mobile Battery Recharging in Swarm Robotics,» chez *International Conference on Bioinspired Methods and Their Applications*, Cham, 2018.

[224] R. K. Ramachandran, Z. Kakish et S. Berman, «Information correlated L\'evy walk exploration and distributed mapping using a swarm of robots,» *arXiv preprint arXiv:1903.04836,* 2019.

[225] K. Lerman et A. Galstyan, «Mathematical model of foraging in a group of robots:Effect of interference,» *Autonomous Robots,* vol. 13, pp. 127-141, 2002.

[226] M. S. Talamali, T. Bose, M. Haire, X. Xu, J. A. Marshall, A. Reina, ... et F. Canciani, «Sophisticated Collective Foraging with Minimalist Agents: A Swarm Robotics Test,» *Swarm Intelligence,* vol. 10349, pp. 144-156, 2019.

[227] S. Garnier, F. Tache, M. Combe, A. Grimal et G. Theraulaz, «Alice in pheromone land: An experimental setup for the study of ant-like robots,» chez *IEEE Swarm Intelligence Symposium*, Washington, DC, USA, 2007.

[228] C. Moeslinger, T. Schmickl et K. Crailsheim, «A minimalist flocking algorithm for swarm robots,» chez *European Conference on Artificial Life*, 2009.

[229] H. Zhao, H. Liu, Y. W. Leung et X. Chu, «Self-adaptive collective motion of swarm robots,» *IEEE Transactions on Automation Science and Engineering,* vol. 15, n°4, pp. 1533-1545, 2018.

[230] A. Chatty, I. Kallel, P. Gaussier et A. M. Alimi, «Emergent complex behaviors for swarm robotic systems by local rules,» chez *IEEE Workshop on Robotic Intelligence In Informationally Structured Space*, Paris, France, 2011.

[231] M. Gauci, J. Chen, W. Li, T. J. Dodd et R. Groß, «Clustering objects with robots that do not compute,» chez *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, Paris, France, 2014.

[232] C. Melhuish, M. Wilson et A. Sendova-Franks, «Patch sorting: Multi-object clustering using minimalist robots,» chez *European Conference on Artificial Life* , 2001.

[233] A. Vardy, «Accelerated patch sorting by a robotic swarm,» chez *Ninth Conference on Computer and Robot Vision*, 2012.

[234] A. Vardy, G. Vorobyev et W. Banzhaf, «Cache consensus: rapid object sorting by a robotic swarm,» *Swarm Intelligence,* vol. 8, n°1, pp. 61-87, 2014.

[235] M. Wilson, C. Melhuish, A. B. Sendova-Franks et S. Scholes, «Algorithms for building annular structures with minimalist robots inspired by brood sorting in ant colonies,» *Autonomous Robots,* vol. 17, p. 115–136, 2004.

[236] R. Groß, S. Magnenat et F. Mondada, «Segregation in swarms of mobile robots based on the brazil nut effect,» chez *IEEE/RSJ International Conference on Intelligent Robots and Systems* , 2009.

[237] J. Chen, M. Gauci, M. J. Price et R. Groß, «Segregation in swarms of e-puck robots based on the brazil nut effect,» chez *Proceedings of the 11th International Con-*, Valencia, Spain, 2012.

[238] G. Carabin, E. Wehrle et R. Vidoni, «A review on energy-saving optimization methods for robotic and automatic systems,» *Robotics,* vol. 6, n°4, pp. 39-60, 2017.

[239] G. Hirzinger, N. Sporer, A. Albu-Schäffer, M. Hähnle, R. Krenn, A. Pascucci et M. Schedl, «DLR's torque-controlled light weight robot III - Are we reaching the technological limits now?,» chez *Proceedings of the IEEE International Conference on Robotics and Automation*, Washington, DC, USA, 2002.

[240] D. Meike et I. Rankis, «New type of power converter for common-ground DC bus sharing to increase the energy efficiency in drive systems,» chez *IEEE International Energy Conference and Exhibition (ENERGYCON)*, Florence, Italy, 2012.

[241] O. Wigström et B. Lennartson, « Energy optimization of trajectories for high level scheduling,» chez *IEEE International Conference on Automation Science and Engineering*, Trieste, Italy, 2011.

[242] J. P. Barreto, F. F. Schöler et B. Corves, «The concept of natural motion for pick and place operations,» *New advances in mechanisms, mechanical transmissions and robotics,* vol. 46, n°1, pp. 89-98, 2017.

[243] P. Boscariol, G. Carabin, A. Gasparetto, N. Lever et R. Vidoni, «Energy-efficient point-to-point trajectory generation for industrial robotic machines,» chez *Proceedings of the ECCOMAS Thematic Conference on Multibody Dynamics*, Barcelona, Spain, 2015.

[244] A. R. Ismail, J. D. Bjerknes, J. Timmis et A. Winfield, «An artificial immune system for self-healing in swarm robotic systems,» chez *International Conference on Information Processing in Cells and Tissues*, Cham, 2015.

[245] X. Zhou et D. Kinny, «Energy-Based Particle Swarm Optimization: Collective Energy Homeostasis in Social Autonomous Robots,» chez *Proceedings of the 2013 IEEE/WIC/ACM*

International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT), Washington, DC, USA , 2013.

[246] J. H. Lee, C. W. Ahn et J. An, «A honey bee swarm-inspired cooperation algorithm for foraging swarm robots: An empirical analysis,» chez *Advanced Intelligent Mechatronics (AIM), 2013 IEEE/ASME International Conference on*, Wollongong, Australia, 2013.

[247] S. Mostaghim, C. Steup et F. Witt, «Energy aware particle swarm optimization as search mechanism for aerial micro-robots,» chez *Computational Intelligence (SSCI), 2016 IEEE Symposium Series on*, Athens, Greece, 2016.

[248] G. Caprari, T. T. Estier et R. Siegwart, «Fascination of down scaling - alice thesugar cube robot,» *Journal of Micromechatronics,* vol. 1, n°3, p. 177–189, 2002.

[249] S. Rutishauser, N. Correll et A. Martinoli, «Collaborative coverage using a swarm of networked miniature robots,» *Robotics and Autonomous Systems,* vol. 57, n°5, pp. 517-525, 2009.

[250] F. Arvin, K. Samsudin et A. R. Ramli, «Development of a miniature robot for swarm robotic application,» *International Journal of Computer and Electrical Engineering,* vol. 1, n°4, pp. 436-442, 2009.

[251] J. Klingner, A. Kanakia, N. Farrow, D. Reishus et N. Correll, «A stick-slip omnidirectional powertrain for low-cost swarm robotics: mechanism, calibration, and control,» chez *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Chicago, Illinois, USA, 2014.

[252] Y. Li, J. Klingner et N. Correll, «Distributed camouflage for swarm robotics and smart materials,» *Autonomous Robots,* vol. 42, n°8, pp. 1635-1650, 2018.

[253] R. Boyle, «Ping pong ball-sized robots can swarm together to form a smart liquid,» *Popular Science,* vol. 12, 2012.

[254] F. B. M. Mondada, X. Raemy, J. Pugh, C. Cianci, A. Klaptocz et A. Martinoli, «The e-puck, a robot designed for education in engineering,» chez *Proceedings of the 9th conference on autonomous robot systems and competitions*, Castelo Branco, Portugal, 2009.

[255] M. Rubenstein, C. Ahler, N. Hoff, A. Cabrera et R. Nagpal, «Kilobot: A low cost robot with scalable operations designed for collective behaviors,» *Robotics and Autonomous Systems,* vol. 62, n°7, pp. 966-975, 2014.

[256] M. Rubenstein, A. Cornejo et R. Nagpal, «Programmable self-assembly in a thousand-robot swarm,» *Science,* vol. 345, n°6198, pp. 795-799, 2014.

[257] G. Valentini, E. Ferrante, H. Hamann et M. Dorigo, «Collective decision with 100 Kilobots: Speed versus accuracy in binary discrimination problems,» *Autonomous Agents and Multi-Agent Systems,* vol. 30, n°3, pp. 553-580, 2016.

[258] M. Rubenstein, A. Cabrera, J. Werfel, G. Habibi, J. McLurkin et R. Nagpal, «Collective transport of complex objects by simple robots: theory and experiments,» chez *Proceedings of*

the 2013 international conference on Autonomous agents and multi-agent systems, St. Paul, MN, USA , 2013.

[259] A. Özgür, S. Lemaignan, W. Johal, M. Beltran, M. Briod, L. Pereyre, P. Dillenbourg et e. al., «Cellulo: Versatile handheld robots for education,» chez *12th ACM/IEEE International Conference on Human-Robot Interaction* , Vienna, Austria, 2017.

[260] F. Arvin, C. Xiong et S. Yue, «Colias-φ: an autonomous micro robot for artificial pheromone communication,» *International Journal of Mechanical Engineering and Robotics Research,* vol. 4, n°4, pp. 349-353, 2015.

[261] J. M. Soares, I. Navarro et A. Martinoli, «The Khepera IV mobile robot: performance evaluation, sensory data and software toolbox,» chez *Robot 2015: Second Iberian Robotics Conference*, Lisbon, Portugal, 2016.

[262] F. Arvin, J. Espinosa, B. Bird, A. West, S. Watson et B. Lennox, «Mona: an affordable mobile robot for swarm robotic applications,» chez *The first UK-RAS Conference on Robotics and Autonomous Systems*, Bristol, UK, 2017.

[263] J. McLurkin, A. J. Lynch, S. Rixner, T. W. Barr, A. Chou, K. Foster et S. Bilstein, «A low-cost multi-robot system for research, teaching, and outreach,» chez *Distributed Autonomous Robotic Systems*, vol. 83, 2013, pp. 597-609.

[264] E. Sklar, «NetLogo, a Multi-agent Simulation Environment,» *Artificial Life,* vol. 13, n°3, pp. 303-311, 2007.

[265] C. Pinciroli, V. Trianni, R. O'Grady, G. Pini, A. Brutschy, M. Brambilla, M. Birattari et e. al., «ARGoS: a modular, parallel, multi-engine simulator for multi-robot systems,» *Swarm intelligence,* vol. 6, n°4, pp. 271-295, 2012.

[266] S. Magnenat, M. Waibel et A. Beyeler, «Enki: The fast 2D robot simulator,» 2011. [En ligne]. Available: http://home.gna.org/enki.

[267] N. Bredeche, J. M. Montanier, B. Weel et E. Haasdijk, «Roborobo! a fast robot simulator for swarm and collective robotics,» *arXiv preprint arXiv:1304.2888,* 2013.

[268] E. Rohmer, S. P. Singh et M. Freese, «V-REP: A versatile and scalable robot simulation framework,» chez *IEEE/RSJ International Conference on Intelligent Robots and Systems*, Tokyo, Japan, 2013.

[269] O. Michel, «Cyberbotics Ltd. Webots™: professional mobile robot simulation,» *International Journal of Advanced Robotic Systems,* vol. 1, n°1, pp. 39-42, 2004.

[270] S. Dähling, L. Razik et A. Monti, «Enabling scalable and fault-tolerant multi-agent systems by utilizing cloud-native computing,» *Autonomous Agents and Multi-Agent Systems,* vol. 35, n°1, pp. 1-27, 2021.

[271] A. Mustafa, M. N. ul Islam et S. Ahmed, «Average Convergence for Directed & Undirected Graphs in Distributed Systems,» *COMPUTER SYSTEMS SCIENCE AND ENGINEERING,* vol. 37, n°3, pp. 399-413, 2021.

[272] W. Fan, P. Chen, D. Shi, X. Guo et L. Kou, «Multi-agent modeling and simulation in the AI age,» *Tsinghua Science and Technology,* vol. 26, n°5, pp. 608-624, 2021.

[273] B. Khaldi et F. Cherif, «An overview of swarm robotics: Swarm intelligence applied to multi-robotics,» *International Journal of Computer Applications,* vol. 126, n°2, pp. 31-37, 2015.

[274] E. Şahin et A. Winfield, «Special issue on swarm robotics,» *Swarm Intelligence,* vol. 2, n°2, pp. 69-72, 2008.

[275] S. Chattunyakit, T. Kondo, I. Nilkhamhang, T. Phatrapornnant et I. Kumazawa, «Two foraging algorithms for a limited number of swarm robots,» chez *The SICE Annual Conference 2013* , Nagoya, Japan, 2013.

[276] G. Mermoud, U. Upadhyay, W. C. Evans et A. Martinoli, «Top-down vs. bottom-up model-based methodologies for distributed control: a comparative experimental study,» chez *Experimental Robotics*, Berlin, Heidelberg, Springer, 2014, pp. 615-629.

[277] V. Trianni et M. López-Ibáñez, «Advantages of task-specific multi-objective optimisation in evolutionary robotics,» *PloS one,* vol. 10, n°8, p. e0136406, 2015.

[278] A. Konak, D. W. Coit et A. E. Smith, «Multi-objective optimization using genetic algorithms: A tutorial,» *Reliability Engineering & System Safety,* vol. 91, n°9, pp. 992-1007, 2006.

[279] M. Amos et O. Don, «Swarm-based spatial sorting,» *International journal of intelligent computing and cybernetics,* vol. 1, n°3, pp. 454-473, 2008.

[280] M. Gauci, J. Chen, W. Li, T. J. Dodd et R. Groß, «Clustering objects with robots that do not compute,» chez *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, Paris, France, 2014.

[281] R. Rao, «Teaching-Learning-Based Optimization Algorithm,» chez *Teaching Learning Based Optimization Algorithm*, Cham, Springer, 2016, pp. 9-39.

[282] T. Lust et J. Teghem, «The multiobjective multidimensional knapsack problem: a survey and a new approach,» *International Transactions in Operational Research,* vol. 19, n° 4, pp. 495-520, 2012.

Annexes

Annexe 1. Exemple d'un POMO non-linière continu

Soit le problème multi-objectif non-linéaire continu suivant :

$$\min_{x} \begin{cases} F_1(x) = 3(x_1 - 3)^2 + 4(x_2 - 1)^2 \\ F_2(x) = (x_1 - 8)^2 + (x_2 + 2)^2 \end{cases}$$

$$\text{Sujet aux Contraintes : } \begin{cases} g_1(x) = (x_1 - 5)^2 + {x_2}^2 - 25 \leq 0 \\ g_2(x) = (x_1 - 8)^2 - (x_2 + 3)^2 - 0.2 \leq 0 \\ 0 \leq x_1 \leq 10 \\ -5 \leq x_2 \leq 5 \end{cases}$$

Les illustrations suivantes sont le résultat d'implémentation de cet exemple dans MATLAB. Dans la figure A1.1, on présente l'espace de décision avec deux variables x_1 *et* x_2, les contours de F_1 *et* F_2 et les contours de contraintes sur les bordures $g_1 = 0$ *et* $g_2 = 0$. Les valeurs de variables de décision satisfaisant les contraintes, qui forment l'espace de décision indiqué par X dans la figure, sont délimitées par les deux contraintes.

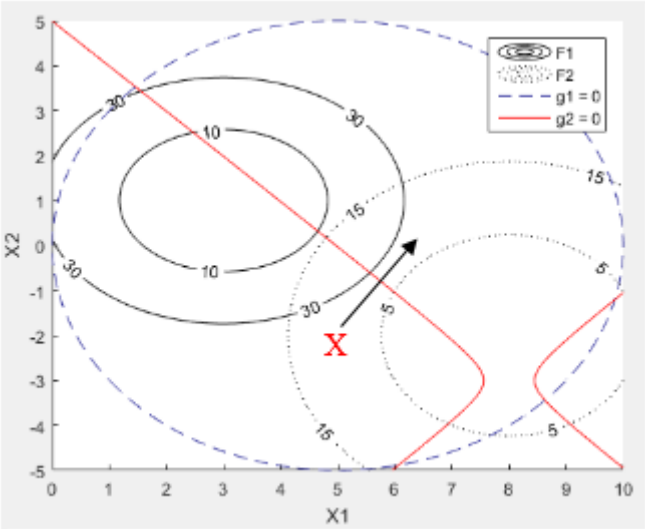


Figure A1.1 : L'espace de décision avec quelques contours

La figure A1.2, illustre graphiquement les deux espaces X et Z de l'exemple donné ci-dessus. L'espace de décision X est délimité par les intervalles de x_1 *et* x_2 et les contraintes d'inégalité $g_1 et\ g_2$, alors que l'espace d'objectifs Z représente les images f_1 *et* f_2 des variables satisfaisant les contraintes du problème.

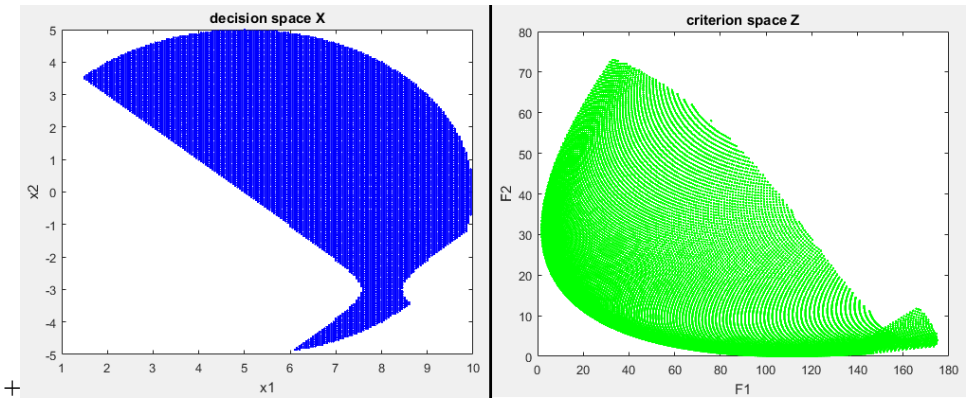


Figure A1.2 : Illustration de l'espace de décision et l'espace d'objectifs de l'exemple

Annexe 2. Exemple d'un POMO combinatoire : Sac à dos multi-objectif multidimensionnel

Plusieurs problèmes réels tels que la répartition des budgets et l'allocation de ressources peuvent être modélisés par le modèle « ***sac à dos*** » dont l'idée du problème mono-objectif est d'emporter des équipements dans un sac à dos avec un poids total limité et fixé par une borne supérieure W, tout en maximisant le bénéfice total $f(x)$. Chaque objet O_i est caractérisé par un poids w_i et un bénéfice c_i. Pour illustrer concrètement cela, prenons l'exemple du transport de marchandises dans des conteneurs, avec une capacité maximale prédéfinie. Les produits à charger dans le conteneur sont modélisés par des objets ; où chaque produit a un poids et un avantage différents des autres. Ce problème est modélisé par le sac à dos afin de maximiser le profit total du commerçant tout en respectant la capacité maximale du conteneur. Il faut noter que $\sum_{i=1}^{n} w_i > W$, ce qui laisse entendre qu'on ne peut pas mettre tous les objets dans le sac à dos. Ce problème mono-objectif combinatoire est considéré comme un problème NP-difficile [282]. Sa formulation mono-objectif est donnée par :

$$\max_x f(x) = \sum_{i=1}^{n} c_i x_i$$

Sujet à : $\sum_{i=1}^{n} w_i x_i \leq W$

$x_i \epsilon \{0,1\}\ i = 1, \dots, n$, n est le nombre d'objets

$$\text{Tel que} \begin{cases} x_i = 1 & \text{si l'objet est retenu} \\ x_i = 0 & sinon \end{cases}$$

Le problème du sac à dos multi-objectif multidimensionnel (MOMKP) est formulé comme suit :

$$\max_x f_k(x) = \sum_{i=1}^{n} c_k^i x_i \qquad k = 1, \dots, p$$

Sujet à : $\sum_{i=1}^{n} w_i^j x_i \leq W_j \qquad j = 1, \dots, m$

$$x_i \epsilon \{0,1\}\ i = 1, \dots, n$$

Tel que p est le nombre de fonctions objectifs (bénéfices), m le nombre de caractéristiques w_i^j de chaque objet O_i (poids, volume, etc.), n est le nombre d'objets. Aussi, dans ce cas $\sum_{i=1}^{n} w_i^j > W_j \qquad j = 1, \dots, m$. Chaque objet possède différents bénéfices pour chaque objectif $c_k^i\ tel\ que\ k = 1, \dots, p$. Dans le cas où $m = 1$ le problème est dit multi-objectif (MOKP) [282].

Les caractéristiques des sacs à dos et d'objets de l'exemple expérimental donné sont résumées, respectivement, dans les tableaux A2.1 et A2.2.

Tableau A2.1. Les caractéristiques des sacs à dos

	Sac à dos 1	Sac à dos 2
Poids	40	36
Volume	36	30

Tableau A2.2. Les caractéristiques des dix objets

Objet	w^1 poids	w^2 volume	c^1 benefit 1	c^2 benefit 2
1	6	6	3	14
2	6	3	4	17
3	12	6	11	12
4	2	5	10	4
5	2	4	18	13
6	11	2	16	3
7	10	6	15	3
8	9	4	2	12
9	12	5	3	7
10	11	6	3	12

Tableau A2.3. Les solutions de Pareto

Combinaison d'objets	(f_1, f_2)	$(\sum_{i=1}^{`} w^1, \sum_{i=1}^{`} w^2)$
2 4 5 6 7	(63,40)	(31,20)
2 3 4 5 6	(59,49)	(33,20)
1 2 5 6 7	(56,50)	(35,21)
1 2 4 5 6 8	(53,63)	(36,24)
1 2 4 5 8 10	(40,72)	(36,28)

Avec ces données, 1021 combinaisons d'objets sont possibles, parmi elles, 399 combinaisons vérifient les contraintes de poids et de volume. Les valeurs des deux objectifs de ces 399 points sont présentées sur la figure A2.1, parmi ces solutions, 5 solutions non-dominées sont entourées en rouge. Les combinaisons d'objets donnant ces solutions, leurs (f_1, f_2) et leurs sommes de poids et leur volume sont résumés dans le tableau A2.3.

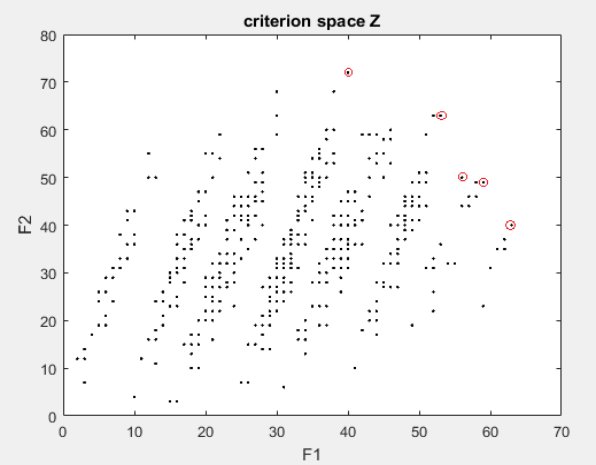


Figure A2.1 : Illustration de l'espace d'objectifs et le front de Pareto du problème de sac à dos

Annexe 3. Les procédures de comportements d'agent-robot

Algorithme A3.1 : Déplacer

Les entrées : La situation actuelle **S : liste ;** le comportement à exécuter **C ∈{0, 1, 2} ;**
Les sorties : Changement possible de la position et/ou la direction ;
Changement de l'énergie d'agent-robot ;

```
01  Si S [0] = 0 et C = 0 // la cellule 0 est vide et le comportement à exécuter est un déplacement
02  vers cette cellule// Alors Avancer ; Energie = Energie - 3 ;
03  Sinon
04    Si S [1] = 0 et C = 1 Alors Tourner à droite 90° ; Avancer ; Energie = Energie - 4 ;
05    Sinon
06      Si S [2] = 0 et C = 1 Alors Tourner à gauche 90° ; Avancer ; Energie = Energie - 4 ;
07      Sinon
08        Si S [3] = 0 et C = 2 Alors // chercher un chemin
09          Si S [0] = 0 Alors Avancer ; Tourner à droite 90° ; Avancer ; Energie = Energie - 7 ;
10          Sinon
11            Si S [1] = 0 Alors Tourner 90° à droite ; Avancer ; Tourner à gauche 90° ; Avancer ;
12              Energie = Energie – 8 ;
13            Sinon // les deux chemins sont bloqués
14              Tourner 180° ; Energie = Energie – 2 ;
15            Fin si
16          Fin si
17        Sinon
18          Si S [4] = 0 et C = 2 Alors // chercher un chemin
19          Si S [0] = 0 Alors Avancer ; Tourner à gauche 90° ; Avancer ; Energie = Energie - 7 ;
20          Sinon
21            Si S [2] = 0 Alors Tourner 90° à gauche ; Avancer ; Tourner à droite 90° ; Avancer ;
22              Energie = Energie – 8 ;
23            Sinon // les deux chemins sont bloqués
24              Tourner 180° ; Energie = Energie – 2 ;
25            Fin si
26          Fin si
27        Sinon // le comportement ne s'exécute pas avec une telle perception
28          Tourner 180° ; Energie = Energie – 2 ;
29        Fin si
30      Fin si
31    Fin si
32  Fin si
33  Fin si
```

Algorithme A3.2 : Ramasser

Les entrées : La situation actuelle **S : liste ;** le comportement à exécuter **C ∈{3, 4, 5} ;**
Les sorties : Changement possible de la position et/ou la direction ;
Changement de l'énergie d'agent-robot ;
Et ramassage possible d'un objet ;

```
01  Si S [0] = 2 et C = 3 // la cellule 0 contient un objet et le comportement à exécuter est ramasser
02  un objet de cette cellule// Alors Avancer ; Ramasser ; Energie = Energie - 7 ;
03  Sinon
04    Si S [1] = 2 et C = 4 Alors Tourner à droite 90° ; Avancer ; Ramasser ; Energie = Energie - 8 ;
05    Sinon
06      Si S [2] = 2 et C = 4 Alors Tourner à gauche 90° ; Avancer ; Ramasser ;
07          Energie = Energie - 8 ;
08      Sinon
09        Si S [3] = 2 et C = 5 Alors // chercher un chemin
```

```
10          Si S [0] = 0 Alors Avancer ; Tourner à droite 90° ; Avancer ; Ramasser ;
11             Energie = Energie - 11 ;
12          Sinon
13            Si S [1] = 0 Alors Tourner 90° à droite ; Avancer ; Tourner à gauche 90° ; Avancer ;
14               Ramasser ; Energie = Energie – 12 ;
15            Sinon // les deux chemins sont bloqués
16              Tourner 180° ; Energie = Energie – 2 ;
17            Fin si
18          Fin si
19        Sinon
20          Si S [4] = 2 et C = 5 Alors // chercher un chemin
21            Si S [0] = 0 Alors Avancer ; Tourner à gauche 90° ; Avancer ; Ramasser ;
22               Energie = Energie - 11 ;
23          Sinon
24            Si S [2] = 0 Alors Tourner 90° à gauche ; Avancer ; Tourner à droite 90° ; Avancer ;
25               Ramasser ; Energie = Energie – 12 ;
26            Sinon // les deux chemins sont bloqués
27              Tourner 180° ; Energie = Energie – 2 ;
28            Fin si
29          Fin si
30          Sinon // le comportement ne s'exécute pas avec une telle perception
31            Tourner 180° ; Energie = Energie – 2 ;
32        Fin si
33      Fin si
34    Fin si
35  Fin si
36 Fin si
```

Algorithme A3.3 : Déposer

Les entrées : La situation actuelle **S : liste ;** le comportement à exécuter **C ∈{3, 4, 5} ;**
Les sorties : Changement possible de la direction ; Changement de l'énergie d'agent-robot ;
Et dépôt possible d'un objet ;

```
01  Si S [0] = 0 et C = 3 // la cellule 0 est vide et le comportement à exécuter est un dépôt de l'objet
02  dans cette cellule// Alors Avancer ; Déposer ; Tourner 180° ; Avancer ; Energie = Energie - 12 ;
03  Sinon
04    Si S [1] = 0 et C = 4 Alors Tourner à droite 90° ; Avancer ; Déposer ; Tourner 180° ; Avancer ;
05       Energie = Energie - 13 ;
06    Sinon
07      Si S [2] = 0 et C = 4 Alors Tourner à gauche 90° ; Avancer ; Déposer ; Tourner 180° ;
08         Avancer ; Energie = Energie - 13 ;
09      Sinon
10        Si S [3] = 0 et C = 5 Alors // chercher un chemin
11          Si S [0] = 0 Alors Avancer ; Tourner à droite 90° ; Avancer ; Déposer ; Tourner 180° ;
12             Avancer ; Tourner à gauche 90° ; Avancer ; Energie = Energie - 20 ;
13          Sinon
14            Si S [1] = 0 Alors Tourner 90° à droite ; Avancer ; Tourner à gauche 90° ; Avancer ;
15               Déposer ; Tourner 180° ; Avancer ; Tourner à droite 90° ; Avancer ;
16               Energie = Energie – 21 ;
17            Sinon // les deux chemins sont bloqués
18              Tourner 180° ; Energie = Energie – 2 ;
19            Fin si
20          Fin si
21        Sinon
22          Si S [4] = 0 et C = 5 Alors // chercher un chemin
23            Si S [0] = 0 Alors Avancer ; Tourner à gauche 90° ; Avancer ; Déposer ; Tourner 180° ;
```

```
24              Avancer ; Tourner à droite 90° ; Avancer ; Energie = Energie - 20 ;
25            Sinon
26              Si S [2] = 0 Alors Tourner 90° à gauche ; Avancer ; Tourner à droite 90° ; Avancer ;
27                 Déposer ; Tourner 180° ; Avancer ; Tourner à gauche 90° ; Avancer ;
28                 Energie = Energie – 21 ;
29              Sinon // les deux chemins sont bloqués
30                Tourner 180° ; Energie = Energie – 2 ;
31              Fin si
32            Fin si
33          Sinon // le comportement ne s'exécute pas avec une telle perception
34            Tourner 180° ; Energie = Energie – 2 ;
35          Fin si
36        Fin si
27      Fin si
38    Fin si
39  Fin si
```

Annexe 4. Les Procédures connexes à l'algorithme NSGA-II

Algorithme A4.1 : Non-dominated Sorting

Les entrées : L1 : **l**iste d'individus ; // contient les ID des individus
Les sorties : L1 : liste d'individus avec un changement dans Rank, Dominationset, Dominatedcount des individus ;
Front : liste ;

```
Front = {} ;            // Front de Pareto
Pour chaque individu
  Rank = 0, Dominationset = {}, Dominatedcount = 0 ;
Fin pour
Pour i = 1 à taille (L1)
  Pour j = i+1 à taille (L1)
    Si individu L1(i) domine individu L1(j) Alors
      Individu L1(i) : Dominationset = Dominationset + {L1(j)} ;
      Individu L1(j) : Dominatedcount = Dominatedcount + 1 ;
    Fin si
    Si individu L1(j) domine individu L1(i) Alors
      Individu L1(i) : Dominatedcount = Dominatedcount + 1 ;
      Individu L1(j) : Dominationset = Dominationset + {L1(i)} ;
    Fin si
  Fin pour
 Individu i : Si Dominatedcount = 0 Alors Rank = 1 ; Front = Front + {L1(i)} ;
Fin pour
Rank-pop = individus ayant Rank = 1 ;
 R = 2 ;
Tant qu'il y a des individus avec Rank = 0
  Pour chaque individu dans Rank-pop :
    Pour chaque individu dans la liste Dominationset : Dominatedcount = Dominatedcount - 1 ;
  Les individus ayant Dominatedcount = 0 et Rank = 0 : Rank = R ;
  Rank-pop = individus ayant Rank = R ;
  R = R + 1 ;
Fin tant que
```

Algorithme A4.2 : Crowding distance

Les entrées : L1 : liste d'individus ;
Les sorties : L1 : liste d'individus avec un changement dans Crowddist des individus ;

```
Pour chaque individu
 Crowddist = 0.0 ;
Fin pour
Pour chaque front          // les individus ayant le même Rank
  Tri croissant des individus de ce front selon F_q ;
   Les deux individus d'extrémités :  Crowddist F_q = 1.0 ;
   Pour chacun des autres individus : Crowddist F_q est calculée en utilisant l'équation4.5
  Tri croissant des individus de ce front selon F_e ;
   Les deux individus d'extrémités :  Crowddist = Crowddist F_q + 1.0 ;
   Pour chacun des autres individus : Crowddist = Crowddist F_q + Crowddist F_e (calculée en
utilisant l'équation4.5)
Fin pour
```

Algorithme A4.3 : Génération d'une nouvelle population

Les entrées : L1 : liste d'individus ; nCrois, nMut, mu : réels ;
Les sorties : L2 : liste d'individus ;

```
01  // croisement
02  Parents = Sélection aléatoire (L1, nCrois) ;
03  Diviser la liste Parents en P1 et P2 ;
04  Pour i = 1 à taille (P1)
05    [Chrom1, Chrom2] = Croissement (Chrom d'individu P1(i), Chrom d'individu P2(i));
06    Créer deux nouveaux individus I1, I2 ;
07    Individu I1 : Chrom = Chrom1 ; ajouter son ID dans la liste L2 ;
08    Individu I2 : Chrom = Chrom2 ; ajouter son ID dans la liste L2 ;
09  Fin pour
10  // mutation
11  Mutants = Sélection aléatoire (L1, nMut) ;
12  Pour i = 1 à nMut
13    [Chrom1] = Mutation (Chrom d'individu Mutants(i), mu) ;        // Algorithme A4.5
14    Créer un nouveau individu I ;
15    Individu I : Chrom = Chrom1 ; ajouter son ID dans la liste L2 ;
16  Fin pour
```

Algorithme A4.4 : Le croisement

Les entrées : Chrom1, Chrom2 : tableaux ;
Les sorties : Chrom1, Chrom2 : tableaux ;

```
01  Choisir un point de scission aléatoire entre 2 et 485
02  Découper Chrom1 en X1 et Y1 dans point de scission ;
03  Découper Chrom 2 en X2 et Y2 point de scission ;
04  Chrom1 = X1 + Y2 ;
05  Chrom 2 = X2 + Y1 ;
```

Algorithme A4.5 : La mutation

Les entrées : Chrom : tableau ; mu : réel ;
Les sorties : Chrom : tableau ;

```
01   [N-mut] = entier (mu * 486) ;  // calculer le nombre de gènes à muter
02  Sélectionner aléatoirement N-mut gènes parmi les 486 du Chrom ;
03  Modifier chaque gène sélectionné selon le schéma à droite ;
```