



République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université des Sciences et de la Technologie Houari Boumediene

Faculté d'Electronique et d'Informatique
Département Informatique

Mémoire de Master

Filière: Informatique

Spécialité:
Système Informatique Intelligent

Thème

Développement d'un système multi-robots basé sur l'intelligence en essaim pour la détection de multiples cibles dans un environnement complexe et inconnu.

Sujet Proposé et encadré par:

Pr. Habiba DRIAS

Soutenu le : 13/06/2019

Réalisation:

MOULAI Hassina Safaa

HOUACINE Naila Aziza

Devant le jury composé de:

Dr DAOUDI Mourad Président
Dr KHENNAK Ilyes Membre

Binôme N° : 81/ 2019

Remerciements

Je tiens à adresser mes remerciements au bon Dieu pour m'avoir donné le courage et patience de mener à terme ce modeste travail.

Je voudrais dans un second temps remercier, notre directrice de mémoire Professeur Habiba DRIAS, notre enseignante aussi durant notre formation, pour sa patience, sa disponibilité et surtout ses judicieux conseils, qui ont contribué à alimenter ma réflexion, ainsi que ces enseignements qui nous ont instruit de plus d'alimenter notre savoir sur plusieurs plans.

Je remercie également messieurs les membres du jury, Mr Mourad DAOUDI et Mr Ilyes KHENNAK d'avoir accepté d'examiner et de juger notre travail.

Je tiens particulièrement à témoigner une reconnaissance et gratitude envers tous nos professeurs qui ont contribué à notre formation particulièrement celle de master qui nous a formé, forgé notre caractère et nous a poussé à travailler davantage.

Un grand merci à tous mes amis qui ont été présents par leurs conseils et leur motivation tout au long de ce projet, particulièrement ma binôme Naila pour sa patience et son sérieux.

Enfin, mes plus chers remerciements vont à mes parents et membre de ma famille pour leur présence et encouragement et leur soutien inestimable et bien plus encore que je ne saurai citer.

Remerciements

En tout premier lieu, je remercie ALLAH le tout-puissant, de m'avoir donné la force, l'audace, mais aussi la patience et le courage de persévérer et de mener à terme ce travail qui me tenait tant à cœur.

J'adresse mes sincères remerciements et toute ma gratitude à notre enseignante et promotrice Mme Habiba DRIAS, que ce soit pour les trois semestres d'enseignement durant lesquels elle n'a cessé de nous encourager, de nourrir nos esprits et de nous pousser à donner le meilleur de nous même, que pour son accompagnement tout au long de ces 4 mois de travail sur ce projet, de par sa disponibilité, ses conseils et orientations irremplaçables, mais aussi pour l'autonomie et la confiance qu'elle nous a accordée, j'ai grandi à vos côtés, merci.

J'adresse également, mes remerciements à Mr Mourad DAOUDI qui nous fait l'honneur de présider le jury de notre soutenance. Mes remerciements vont aussi à Mr Ilyes KHEN-NAK pour avoir accepté de faire partie de ce jury.

Aussi, toute ma gratitude va à tous les enseignants de l'USTHB qui non seulement ont alimenté mon esprit et enrichi mon savoir mais aussi forgé mon caractère, mesdames: Khellaf, Aouat, Kadri, Amrouni(Kechid), Aliouane, Challal, Zemmouchi, Mehdi, Had-douche, Amrous, Bouziane, Ourahmoune, Boulekrinat, Gharbi, Zaouche, Chenait, Maha-doui, Abdat, Hank, Baba Ali, Boughaci, Hadjazi, Bellala, Batoul, Mazouzi, Mankour, Rouaiguia, Hadouh, ainsi que messieurs: Guessoum, Azzoune, Kechid, Isli, Boukhelfa, Atif, Smaili, Abada, Djouadi, Djouama, Bouzeghoub, Bahloul, Laichi, Boudjaja, Daoudi, Amani, Ainouz. Et tous les autres que je n'ai pas eus la chance d'avoir comme enseignant(e)s, j'ai pour ambition de revenir en cette faculté qui m'est chère, qui sait peut être nous serons collègues un jour inch'allah.

Aussi loin que remontent mes souvenirs, je me rappelle et me rappellerai toujours de tout ce que les deux êtres que j'admire le plus en ce monde et les plus chers à mes yeux ont fait pour moi et je ne les remercierai jamais assez qu'importe ce que je ferai. Papa, maman merci pour votre amour, votre présence, votre soutien, vos conseils, vos sacrifices, votre éducation et les valeurs que vous m'avez inculquées, tous vos efforts, votre patience, vos encouragements, ... Et tant d'autres choses que je ne pourrais tous les citer. Merci je suis on ne peut plus fière de vous avoir et fière de ce que je suis aujourd'hui grâce à vous, j'espère vous avoir rendu fière à mon tour.

Je remercie chaleureusement mon grand frère pour ses encouragements, son support moral et pour m'avoir toujours guidé et inspiré. Je n'oublierai pas de remercier ma petite sœur qui a toujours cru en moi. Ma pensée va également à ma tendre grand-mère qui ne cesse de prier jour et nuit pour ma réussite et toute ma famille qui m'ont soutenu tout le long de ce projet.

Enfin, je remercie en général tous mes ami(e)s et camarades que je ne pourrais nommer un à un et en particulier mon amie et binôme Hassina Safaa MOULAI, pour leur appui et leur confiance qu'ils m'ont témoignée et pas seulement dans le cadre de ce travail mais durant tout mon cursus, du fond du cœur je vous souhaite à tous, que vos réussites s'enchaînent les unes après les autres. Afin de n'oublier personne, à tous les intervenants de près ou de loin, je présente mes remerciements, mon respect et ma gratitude.

Naila

*Je dédie ce travail à la mémoire de mon grand père allah y rahmou,
à mes parents bien-aimés et à toute ma famille ainsi qu'à
toute l'équipe de Master SII 2017 - 2019.*

Naila

Résumé

La frontière entre la fiction et la réalité devient chaque jour plus mince, due aux évolutions technologiques qui ont émergé au fil des années. Les robots ne font plus partie de la science-fiction seulement, ils se sont installés dans notre quotidien offrant de nouvelles capacités à l'homme pour sa vie de tous les jours. Les systèmes multi-robots ont conquis le monde de la recherche dans multiples domaines grâce à l'intelligence des robots qui est due à leur coopération lors de la planification des tâches. C'est la raison pour laquelle plusieurs études se sont donné comme but de les rendre de plus en plus performants, robustes et efficaces.

L'un de leurs domaines d'application les plus récemment exploitées est la recherche de cibles. C'est un problème qui consiste à maximiser le nombre de cibles trouvées dans une certaine zone ayant une superficie connue et des caractéristiques qui lui sont propres (Présence d'obstacles, nombre de cibles, ..etc). La planification des actions des robots devient alors un défi faisant face à des contraintes liées aux caractéristiques de l'environnement.

Notre projet consiste à apporter des solutions à la recherche de cibles en adaptant des approches d'intelligence en essaim à cette thématique, communément appelées "méta-heuristiques". Ces approches n'ont pas encore été adaptées sur ce type de problème et qui sont : *BSO* et *Multi-BSO* s'inspirant de l'intelligence d'essaim d'abeilles, *EHO* et *ESWSA* basées groupes d'éléphants qui s'inspirent de la mémoire individuelle des éléphants ainsi que l'intelligence des clans. Elles vont devoir trouver un nombre maximum de cibles en évitant les obstacles potentiellement présents dans l'environnement inconnu.

Mots clés: Systèmes Multi-robots, Recherche de cibles, Évitement d'obstacles, Méta-heuristiques, Intelligence en essaim, BSO, EHO, ESWSA, Multi-BSO.

Abstract

The borderline between fiction and reality is getting thinner every day due to technological developments that have emerged over the years. Robots are no longer just part of science fiction. They have become part of our daily lives, offering new abilities to man for his everyday life. Multi-robot systems have conquered the world of research due to the cooperation that makes robots intelligent as a group. Several studies are undertaken toward improving them by making them more robust and efficient. Target search is one of their most recently exploited fields of application. It consists in maximizing the number of targets found in a certain area that we call environment. The planning of robot actions becomes a challenge facing constraints related to the characteristics of the environment. Our project consists in providing solutions for Target search problem by adapting swarm intelligence approaches, commonly called "metaheuristics" which have not yet been adapted to this problem. The first two approaches, BSO and Multi-BSO are inspired by the intelligence of a swarm of bees, while the two last ones EHO and EWSA are elephant group based that draws inspiration from the individual memory of elephants as well as the intelligence of clans. This metaheuristics will aim to find a maximum number of targets by avoiding potentially present obstacles on the unknown environment.

Keywords: Multi-robots system, Target search, Obstacles avoidance, Metaheuristics, Swarm intelligence, BSO,EHO , EWSA, Multi-BSO.

ملخص

أصبح الحد الفاصل بين الخيال والواقع أرق كل يوم بسبب التغيرات التكنولوجية التي ظهرت على مر السنين. لم تعد الروبوتات جزءاً من الخيال العلمي فقط ، بل استقرت في حياتنا اليومية من خلال توفير وسائل راحة جديدة للإنسان طوال حياته اليومية

فقد غزت الأنظمة متعددة الروبوتات عالم البحث بسبب ذكاء الروبوتات كمجموعة وهو ثمرة من تعاونهم. تعمل العديد من الدراسات على تحسينها بجعلها أكثر قوة وفعالية . تعد إشكالية اكتشاف الأشياء المستهدفة واحداً من مجالات التطبيق المستغلة حديثاً ، وهو يهدف إلى إيجاد أكبر عدد ممكن من الأهداف في منطقة معينة نسبياً البيئة ، ان تخطيط تصرفات الروبوت يمثل تحدياً يواجهه قيوداً تتعلق بخصائص البيئة.

لمعالجة هذه الإشكالية ، يقترح مشروعانا حلولاً لمشكلة اكتشاف الأشياء المستهدفة من خلال التقنيات التي تحاول نمذجة الذكاء في سرب والسلوك الذكي للمجموعة. هذه تقنيات لم يتم القيام به بعد لهذا النوع من المشاكل. Swarm - Multi BSO و BSO كلاهما يحاول نسخ السلوك الذكي لسرب النحل.

EHO و EWSA كلاهما يحاول نسخ خصائص الذاكرة الفردية للأفيال وكذلك ذكاء عشرات الأفيال ،

هذه التقنيات يجب أن تجد أكبر عدد ممكن من الأهداف.

مع تجنب العقبات التي قد تكون موجودة في بيئه غير معروفة مسبقاً.

الكلمات الدالة: مشكلة البحث عن الأشياء المستهدفة ، أنظمة متعددة الروبوت ، ذكاء المجموعة ، تجنب العقبات

Contents

Remerciements	iii
Introduction	1
1 Chapitre 1 : État de l'art	3
1.1 Introduction	3
1.2 Problématique	3
1.3 Paramètres du problème	3
1.3.1 Les systèmes multi-robots (SMRs)	5
1.3.1.1 Définition	5
1.3.1.2 Caractéristiques des SMRs [surv1]	5
1.3.1.3 Système de communication [surv2]	5
1.3.1.4 Comportement des robots [surv2]	5
1.3.2 Les capteurs	6
1.3.3 Les cibles	6
1.3.4 L'environnement [surv2]	6
1.3.5 Les obstacles	6
1.4 Travaux connexes	7
1.4.1 Les algorithmes anti-regroupements [Miao2010]	7
1.4.2 Approche des champs de potentiel [surv2]	7
1.4.3 Approches basées intelligence en essaim	7
1.4.3.1 BSO (Bee Swarm Optimization) (2013)	7
1.4.3.2 A-RPSO	8
1.4.3.3 MFSO (Multi-swarm hybrid FOA-PSO)	8
1.5 Conclusion	8
2 Chapitre 2 : Intelligence en essaim : BSO, EHO, EWSWA et GA	9
2.1 Introduction	9
2.2 Les algorithmes basées intelligence en essaim	9
2.3 Bee Swarm Optimization [Drias]	10
2.3.1 Inspiration des abeilles	10
2.3.2 Analogie entre le comportement des abeilles et BSO	11
2.3.3 Paramètres et fonctions	12
2.3.4 Pseudo-code BSO	13
2.4 EHO & EWSWA : Deux algorithmes basés essaim d'éléphants	14
2.4.1 Inspiration des éléphants	14
2.4.2 EHO : Elephant Herding Optimization	15
2.4.2.1 Analogie entre les éléphants et EHO [EHO2]	15
2.4.2.2 Paramètres et fonctions [EHO2]	15
2.4.2.3 Pseudo code de EHO [EHO2]	16
2.4.3 EWSWA : Elephant Swarm Water Search Algorithm	17
2.4.3.1 Analogie entre les éléphants et EWSWA [EHO1]	17
2.4.3.2 Paramètres et fonctions [EHO1]	18
2.4.3.3 Pseudo code de EWSWA [EHO1]	20

2.4.4	Remarque	21
2.5	Algorithme Génétique [GA _{These}]	21
2.5.1	Inspiration de la génétique	21
2.5.2	Métaphore entre l'ADN et un algorithme génétique	21
2.5.3	Paramètres et fonctions	22
2.5.3.1	Opération de croisement	22
2.5.3.2	Opération de mutation	23
2.5.4	Pseudo code (GA incrémental)	24
2.6	Conclusion	24
3	Chapitre 3 : Modélisation du problème de recherche de cibles	26
3.1	Introduction	26
3.2	Modélisation de l'environnement de recherche	26
3.2.1	Représentation de la solution	27
3.2.2	Fonction objectif	27
3.2.3	Les cibles	27
3.2.4	Les obstacles	28
3.3	Stratégies d'évitement d'obstacles	28
3.3.1	Description des approches existantes	29
3.3.1.1	Les algorithmes Bug [Sara, Bug]	29
3.3.1.2	La méthode basée sur les champs de potentiel PF (Potential Field) [Sara]	29
3.3.1.3	La méthode basée sur la fenêtre dynamique DW (Dynamic Window) [Sara]	30
3.3.1.4	La logique floue	30
3.3.1.5	L'échantillonnage de l'espace d'entrée ISS (Input Space Sample)	30
3.3.2	Discussion et choix	31
3.3.3	Paramètres de la méthode d'échantillonnage	31
3.3.3.1	Champ de vision	32
3.3.3.2	Portée locale d'un robot	32
3.3.3.3	Trajectoire	33
3.3.3.4	Critères de choix de la trajectoire	34
3.4	Auto-Paramétrage avec GA	35
3.4.1	Modélisation	35
3.4.1.1	Solution	35
3.4.1.2	Espace des solutions	36
3.4.1.3	Fonction objectif	36
3.4.1.4	Population	36
3.4.1.5	Croisement	36
3.4.1.6	Mutation	37
3.4.2	Fonctionnement	37
3.4.2.1	Sélection des deux meilleures solutions	37
3.4.2.2	Croisement des deux solutions (parents)	37
3.4.2.3	Évaluation des solutions fils	37
3.4.2.4	Mutation des solutions fils	37
3.4.2.5	Évaluation des solutions mutantes	37
3.4.2.6	Injection des solutions résultantes dans la population	37
3.4.2.7	Passage à la génération suivante	38
3.4.2.8	Critère d'arrêt	38
3.5	Conclusion	39

4 Chapitre 4 : BSO pour le problème de recherche de cibles	40
4.1 Introduction	40
4.2 Mono-BSO vs Multi-BSO	40
4.3 Adaptation de BSO pour le problème de la recherche de cibles	40
4.3.1 Solution	40
4.3.2 Fonction objectif	40
4.3.3 Flip	41
4.3.4 MaxChance	41
4.3.5 Table Dance	41
4.3.5.1 Critère de qualité (Best In Quality)	41
4.3.5.2 Critère de Diversité (Best In Diversity)	42
4.4 Fonctionnement du Mono-BSO	42
4.4.1 Initialisation de la solution de référence (Sref)	42
4.4.2 Insertion de Sref dans la liste <i>Tabou</i>	42
4.4.3 Génération des zones de recherche	43
4.4.4 Affectation des zones aux abeilles	43
4.4.5 Recherche locale pour chaque abeille	43
4.4.6 Déplacement des abeilles	43
4.4.7 Insertion des meilleures solutions dans la table <i>Dance</i>	43
4.4.8 Choix de la nouvelle Sref	43
4.4.9 Critère d'arrêt de la recherche	43
4.5 Fonctionnement du Multi-BSO	46
4.5.1 Initialisation des solutions de référence (Srefs)	46
4.5.2 Insertion de Sref dans la Liste Tabou	46
4.5.3 Génération des zones de recherche	46
4.5.4 Affectation de chaque zone à une abeille	46
4.5.5 Recherche locale pour chaque abeille	46
4.5.6 Insertion des solutions dans les tables <i>Dance</i>	46
4.5.7 Test de la 1 ^{ère} condition d'arrêt	46
4.5.8 Choix de la nouvelle solution de référence	47
4.5.9 Déplacement des robots	47
4.5.10 Test de la 2 ^{ème} condition d'arrêt	47
4.6 Conclusion	47
5 Chapitre 5 : Algorithmes des éléphants pour le problème de recherche de cibles	48
5.1 Introduction	48
5.2 Adaptation de EHO pour le problème de la recherche de cibles	48
5.2.1 Solution	48
5.2.2 Fonction objectif	48
5.2.3 Éléphant	48
5.2.4 Clan	49
5.2.5 Chef de clan (Matriarche)	49
5.2.6 Éléphant mâle	50
5.3 Fonctionnement d'EHO	50
5.3.1 Initialisation des positions des clans	51
5.3.2 Initialisation des positions des éléphants	51
5.3.3 Évaluation des solutions	51
5.3.4 Tri des clans	51
5.3.5 Mise à jour des positions des éléphants	51
5.3.6 Déplacement des robots	51
5.3.7 Test de la condition d'arrêt	51

5.4	Adaptation de EWSA pour le problème de la recherche de cibles	53
5.4.1	Solution	53
5.4.2	Fonction objectif	53
5.4.3	Éléphant	53
5.4.4	Outils de mémoire	53
5.4.5	Vélocité	54
5.4.6	Mécanisme de déplacement	54
5.4.6.1	Mise à jour de la vélocité de l'éléphant	54
5.4.6.2	Mise à jour de la position de l'éléphant	55
5.4.6.3	Mise à jour de W^t (poids d'inertie)	55
5.5	Fonctionnement d'ESWSA	56
5.5.1	Initialisation	56
5.5.1.1	Paramètres empiriques	56
5.5.1.2	Positions initiales des éléphants	56
5.5.1.3	Pbest pour chaque éléphant	56
5.5.1.4	Gbest	56
5.5.2	Mise à jour de la vélocité pour chaque éléphant	56
5.5.3	Calcul de la prochaine position de chaque éléphant	56
5.5.4	Déplacement des robots	57
5.5.5	Évaluation des positions des éléphants	57
5.5.6	Mise à jour du Pbest de chaque éléphant	57
5.5.7	Mise à jour de Gbest	57
5.5.8	Mise à jour du W^t (poids d'inertie)	57
5.5.9	Critère d'arrêt de la recherche	57
5.6	Conclusion	58
6	Chapitre 6 : Validation Expérimentale	59
6.1	Introduction	59
6.2	Environnement de développement	59
6.2.1	Matériel	59
6.2.2	Logiciels	60
6.3	Expérimentations relatives aux approches	60
6.3.1	Paramétrage du mini-GA incrémental	60
6.3.2	Paramétrage des approches de recherche de cibles	60
6.3.2.1	Réglage de Paramètres de BSO	61
6.3.2.2	Réglage de Paramètres de Multi-BSO	61
6.3.2.3	Réglage de Paramètres de EHO	61
6.3.2.4	Réglage de Paramètres de EWSA	62
6.4	Expérimentations relatives à l'environnement	63
6.4.1	Organisation des types d'expérimentations	63
6.4.1.1	Organisation des expérimentations par rapport à la portée des cibles	63
6.4.1.2	Organisation des expérimentations par rapport à la taille de l'environnement	64
6.4.1.3	Organisation des expérimentations par rapport au nombre de cibles	65
6.4.2	Expérimentations par rapport à la portée des cibles	66
6.4.2.1	Taux de réussite	66
6.4.2.2	Variation du nombre d'itérations et temps d'exécution	66
6.4.2.3	- Analyse relative à la portée des cibles	71
6.4.3	Expérimentations par rapport à la taille de l'environnement	71

6.4.3.1	Taux de réussite	71
6.4.3.2	Variation du nombre d'itérations et temps d'exécution	73
6.4.3.3	- Analyse relative à la taille des environnements	78
6.4.4	Expérimentations par rapport au nombre de cibles	79
6.4.4.1	Taux de réussite	79
6.4.4.2	Variation du nombre d'itérations et temps d'exécution	79
6.4.4.3	Analyse relative au nombre de cibles	81
6.4.5	Comparaison des types d'environnement	82
6.4.6	Comparaison de nos quatre approches	82
6.5	Simulateur	82
6.5.1	Fonctionnement	82
6.5.2	Interface	83
6.5.2.1	Environnement	84
6.5.2.2	Métaheuristiques	85
6.6	Conclusion	87
	Conclusion générale et perspectives	88

List of Figures

1.1	Schéma résumant les classes de problèmes de détection de cibles [surv2]	4
1.2	Exemple de capture [surv2]	4
1.3	Exemple de patrouille [surv2]	4
2.1	Classification des méta-heuristiques	10
2.2	Croisement en un point.	22
2.3	Croisement en deux points.	22
2.4	Mutation avec inversement.	23
2.5	Mutation avec permutation.	23
3.1	Représentation de l'environnement.	26
3.2	Représentation d'une solution dans notre modélisation.	27
3.3	Représentation de la portée d'une cible dans notre modélisation.	28
3.4	Calcul du chemin du robot par les algorithmes Bug : (a) Bug1, (b) Bug2 [Bug].	29
3.5	Champs de potentiel pour un environnement contenant deux obstacles et une position but.[ImagePF]	30
3.6	L'approche de la fenêtre dynamique montrant les régions admissibles et interdites par rapport à la fenêtre dynamique des vitesses atteignables par le robot [Sara].	30
3.7	Ensemble de trajectoires générées par échantillonnage de l'espace d'entrées [imageEch].	31
3.8	Détermination de l'angle de vue d'un robot dans notre modélisation.	32
3.9	Représentation de la portée locale d'un robot dans notre modélisation.	32
3.10	Sélection des points (cases) d'une droite avec l'algorithme de BRESENHAM.	33
3.11	Choix de la trajectoire à suivre par le robot dans un environnement à obstacle.	34
3.12	Choix de la trajectoire à suivre par le robot dans des situations complexes.	35
3.13	Croisement de deux solutions.	36
3.14	Mutation d'une solution.	37
3.15	Organigramme du mode de fonctionnement de la méthode GA.	38
4.1	Représentation de la méthode de génération de solutions avec le flip.	41
4.2	Méthode de sélection de la meilleure solution en termes de qualité.	41
4.3	Méthode de sélection de la meilleure solution en termes de Diversité.	42
4.4	Organigramme du mode de fonctionnement de l'approche mono-BSO.	44
4.5	Organigramme du mode de fonctionnement de l'approche multi-BSO.	47
5.1	Représentation de la méthode de mise à jour des positions des éléphants.	48
5.2	Représentation de la technique de génération des positions des éléphants d'un même clan.	49
5.3	Représentation de la méthode de mise à jour de la position du meilleur éléphant/robot.	50
5.4	Représentation de la méthode de mise à jour de la position du pire éléphant/robot.	50
5.5	Organigramme du mode de fonctionnement de l'approche EHO.	52

5.6	Représentation de la méthode de mise à jours du Pbest d'un éléphant.	53
5.7	Représentation de la mise à jour de la vélocité d'un éléphant dans l'approche EWSA.	54
5.8	Organigramme du mode de fonctionnement de l'approche EWSA.	58
6.1	ESWSA à l'itération 4	63
6.2	ESWSA à l'itération 8	63
6.3	ESWSA à l'itération 16	63
6.4	Représentation de l'organisation des expérimentations sur la portée des cibles.	64
6.5	Représentation de l'organisation des expérimentations sur la taille des environnements.	65
6.6	Représentation de l'organisation des expérimentations sur le nombre de cibles.	65
6.7	Comparaison de la variation du taux de réussite des algorithmes selon la portée des cibles.	66
6.8	Comparaison de la variation du nombre d'itérations des algorithmes selon la portée de la cible (sans obstacles).	67
6.9	Comparaison de la variation du temps d'exécution des algorithmes selon la portée de la cible (sans obstacles).	67
6.10	Comparaison de la variation du nombre d'itérations des algorithmes selon la portée des cibles (sans obstacles).	68
6.11	Comparaison de la variation du temps d'exécution des algorithmes selon la portée des cibles (sans obstacles).	68
6.12	Comparaison de la variation du nombre d'itérations des algorithmes selon la portée de la cible (avec obstacles).	68
6.13	Comparaison de la variation du temps d'exécution des algorithmes selon la portée de la cible (avec obstacles).	68
6.14	Comparaison de la variation du nombre d'itérations des algorithmes selon la portée des cibles (avec obstacles).	69
6.15	Comparaison de la variation du temps d'exécution des algorithmes selon la portée des cibles. (avec obstacles)	69
6.16	Comparaison de la variation du nombre d'itérations des algorithmes selon la portée de la cible (complexe).	70
6.17	Comparaison de la variation du temps d'exécution des algorithmes selon la portée de la cible (complexe).	70
6.18	Comparaison de la variation du nombre d'itérations des algorithmes selon la portée des cibles (complexe).	70
6.19	Comparaison de la variation du temps d'exécution des algorithmes selon la portée des cibles (complexe).	70
6.20	Comparaison de la variation du taux de réussite des algorithmes selon la taille de l'environnement (sans obstacles).	71
6.21	Comparaison de la variation du taux de réussite des algorithmes selon la taille de l'environnement (sans obstacles).	72
6.22	Comparaison de la variation du taux de réussite des algorithmes selon la taille de l'environnement (avec obstacles).	72
6.23	Comparaison de la variation du taux de réussite des algorithmes selon la taille de l'environnement (avec obstacles).	72
6.24	Comparaison de la variation du taux de réussite des algorithmes selon la taille de l'environnement (complexe).	73

6.25 Comparaison de la variation du taux de réussite des algorithmes selon la taille de l'environnement (complexe).	73
6.26 Comparaison de la variation du nombre d'itérations des algorithmes selon la taille de l'environnement (sans obstacles).	74
6.27 Comparaison de la variation du temps d'exécution des algorithmes selon la taille de l'environnement (sans obstacles).	74
6.28 Comparaison de la variation du nombre d'itérations des algorithmes selon la taille de l'environnement (sans obstacles).	75
6.29 Comparaison de la variation du temps d'exécution des algorithmes selon la taille de l'environnement (sans obstacles).	75
6.30 Comparaison de la variation du nombre d'itérations des algorithmes selon la taille de l'environnement (avec obstacles).	76
6.31 Comparaison de la variation du temps d'exécution des algorithmes selon la taille de l'environnement (avec obstacles).	76
6.32 Comparaison de la variation du nombre d'itérations des algorithmes selon la taille de l'environnement (avec obstacles).	76
6.33 Comparaison de la variation du temps d'exécution des algorithmes selon la taille de l'environnement (avec obstacles).	76
6.34 Comparaison de la variation du nombre d'itérations des algorithmes selon la taille de l'environnement (complexe).	77
6.35 Comparaison de la variation du temps d'exécution des algorithmes selon la taille de l'environnement (complexe).	77
6.36 Comparaison de la variation du nombre d'itérations des algorithmes selon la taille de l'environnement (complexe).	78
6.37 Comparaison de la variation du temps d'exécution des algorithmes selon la taille de l'environnement (complexe).	78
6.38 Comparaison de la variation du taux de réussite des algorithmes selon le nombre de cibles.	79
6.39 Comparaison de la variation du nombre d'itérations des algorithmes selon le nombre de cible (sans obstacles).	79
6.40 Comparaison de la variation du temps d'exécution des algorithmes selon le nombre de cible (sans obstacles).	79
6.41 Comparaison de la variation du nombre d'itérations des algorithmes selon le nombre de cible (avec obstacles).	80
6.42 Comparaison de la variation du temps d'exécution des algorithmes selon le nombre de cible (avec obstacles).	80
6.43 Comparaison de la variation du nombre d'itérations des algorithmes selon le nombre de cible (complexe).	81
6.44 Comparaison de la variation du temps d'exécution des algorithmes selon le nombre de cible (complexe).	81
6.45 Schéma de communication du système multi-threads.	83
6.46 Présentation de l'interface	83
6.47 Présentation de la section Métaheuristiques	84
6.48 Les types d'environnements possibles.	85
6.49 Affichage de l'environnement selon les préférences insérées	85
6.50 Auto-paramétrage avec l'algorithme génétique	86
6.51 Menu de paramétrage	86
6.52 Menu de paramétrage manuel pour BSO	86
6.53 Informations durant l'exécution de BSO	86
6.54 Liste des paramètres pour BSO	86
6.55 Le slider	87

6.56 Exemple d'exécution de BSO.	87
--	----

List of Tables

2.1	Analogie entre les caractéristiques des essaims d'abeilles et BSO.	11
2.2	Analogie entre les caractéristiques des éléphants et EHO.	15
2.3	Analogie entre les caractéristiques des éléphants et EWSA.	18
2.4	Analogie entre les caractéristiques de l'ADN et GA.	22
3.1	Contraintes sur les paramètres des méta-heuristiques.	35
6.1	Meilleurs paramètres pour BSO.	61
6.2	Meilleurs paramètres pour Multi-BSO.	62
6.3	Meilleurs paramètres pour EHO.	62
6.4	Meilleurs paramètres pour EWSA.	63

Liste des abréviations

IA	Inelligence Artificielle
SAT	Satisfaction
SMRs	Système Multi-Robots
ISS	Input Space Sampling
DW	Dinamic Window
FKD	Filtre de Kalman Distribué
PSO	Particle Swarm Optimization
GA	Genetic Algorithm
BSO	Bee Swarm Optimization
EHO	Elephant Herding Optimization
ESWSA	Elephant Swarm Water Search Algorithm
TS	Taboo Search
SS	Scatter Search
GLS	Guided Local Search
MA	Memetic Algorithm
FFA	Firefly Algorithm
ABSO	Advenced Bee Swarm Optimization
CS	Cuckoo Search
FPA	Flower Pollination Algorithm
BBO	Biogeography-Based Optimization
DE	Differential Evolution
ACO	Ant Colony Optimisation
BA	Bat Algorithm
ABC	Artificial Bee Colony Optimisation
BFO	Bacterial Foraging Optimisation
GSO	Glowworm Swarm Optimisation
FOA	Fruit Fly Optimization Algorithm
MSCM	Multi-Scale Cooperative Mutation
Méta	Méta-heuristic
Algo	Algorithm

Introduction générale

Une vague d'avancées technologiques a submergé le monde durant cette dernière décennie, l'intelligence artificielle est au centre de toutes les innovations étant par conséquent le sujet le plus actuel et un critère d'évaluation du développement des pays du monde.

Cet intérêt croissant a permis à des villes intelligentes de voir le jour ainsi que des voitures autonomes ou encore des robots sociaux comme Sophia [sofia].

Le développement connu par l'intelligence artificielle peut être expliqué par des besoins accrus et nécessaires de gain de temps, d'efforts et de mains d'œuvres humaines dans l'accomplissement de tâches quotidiennes.

Dans le but de participer à ce développement, les chercheurs ont essayé de proposer des solutions automatisant certaines tâches quotidiennes coûteuses en temps ou jugées dangereuses pour l'homme. Les systèmes multi-robots font partie des innovations émergentes et solutions proposées pour répondre à ces besoins. Leur usage s'étale sur des domaines d'application des plus intéressants, allant de l'usage domestique au domaine militaire, passant par l'industrie ou encore le domaine de la santé. Ces multiples applications exploitent l'adaptabilité des robots qui sont capables de résoudre divers problèmes [uses].

Notons que la mobilité des robots est l'une de leurs capacités la plus convoitée, que ce soit dans l'assemblage et l'entreposage, le transport industriel, les équipements de nettoyage du sol [novel], l'exploration de planète, la recherche et le secourisme des victimes dans une zone sinistrée ou encore la localisation de mines ou de bombes [Dadgar2016].

De plus, les systèmes multi-robots se caractérisent par leur coopération, celle-ci permet d'établir une meilleure planification des actions et comportements des robots face au monde qui les entoure pour aboutir à un but commun.

Ces systèmes multi-robots ont été particulièrement efficaces lorsqu'il s'agissait de problèmes de recherche ou détection, que ce soit dans des espaces connus ou inconnus. L'objectif consiste à localiser une ou plusieurs cibles se trouvant dans un certain espace qu'on appelle environnement. Une cible au sens large peut représenter des personnes perdues, des mines, de l'or, des puits de pétrole et bien plus encore.

La détection de cibles est un problème complexe dont il découle un certain nombre de problématiques. On cite, l'optimisation des temps et efforts des robots à travers un choix adéquat de structure de coopération. La conception d'un système d'évitement d'obstacles et de navigation fiable est primordial pour garantir des déplacements sans collisions au sein d'environnements inconnus, pouvant contenir des obstacles. Mais encore, les robots doivent adopter des stratégies de recherche intelligentes en raison de leur manque d'information sur l'environnement.

À travers ce travail, nous chercherons à proposer de nouvelles méthodes de résolution pour la détection de cibles en tentant d'apporter des améliorations au niveau des performances, c'est à dire qu'on vise à minimiser l'effectif (le moins de robots possibles)

pour couvrir toute la zone de recherche, cela en trouvant le maximum de cibles.

Notre contribution comportera une adaptation d'un système multi-robots aux quatre approches inspirées de l'intelligence en essaim, à savoir *BSO* et *Multi-BSO* qui sont basées sur le comportement des abeilles lors de la recherche du pollen, puis *EHO* et *ESWSA* qui sont inspirées de l'intelligence des éléphants en termes de communication et planification lors de la recherche d'eau et de nourriture. Un robot se comportera comme une abeille, un essaim d'abeilles ou un éléphant selon l'approche choisie.

Ces quatre approches devront être adaptées à une même modélisation du problème afin de permettre une comparaison effective des résultats obtenus. Pour cette même raison, la stratégie d'évitement d'obstacles sera exploitée et intégrée pour toutes nos mét-heuristiques.

Ce mémoire compte quatre chapitres organisés comme suit :

D'abord, une synthèse sur les maintes variantes du problème et tous les paramètres influant la direction qu'il peut prendre est présentée dans le chapitre 1. Les algorithmes d'intelligence en essaim choisis pour l'élaboration de nos solutions sont décrits dans le chapitre 2. Le chapitre 3 fera l'objet de la modélisation de notre problème et des stratégies d'évitement d'obstacles. Le chapitre 4 comportera les deux premières contributions basées intelligence des essaims d'abeilles à savoir *BSO* et *Multi-BSO*. Deux autres contributions concernant deux algorithmes s'inspirant des éléphants qui sont *EHO* et *ESWSA*, seront présentées dans le chapitre 5.

L'efficacité des approches proposées sera validée à travers plusieurs expérimentations dont les résultats seront exhibés au chapitre 6. Enfin nous terminerons par une conclusion sur l'ensemble du travail effectué ainsi que des perspectives.

Chapitre 1 : État de l'art

1.1 Introduction

La détection et le suivi de cibles englobent une large variété de problèmes décisionnels, tels que la surveillance, la recherche, les patrouilles, l'observation et la poursuite-évasion. Compte tenu du besoin croissant en effectif, en temps et en minimisation des coûts, de nombreuses applications ont été développées notamment dans des domaines comme le commerce et la défense.

Dans ce chapitre, nous nous attelons à fournir une synthèse sur les différents travaux inhérents à la recherche de multiples cibles dans un environnement inconnu et complexe. Au préalable, nous devons introduire certaines définitions et techniques propres à la nature des environnements auxquels nous aurons à faire face, d'autres concernant les méthodes de résolution. On finira par une conclusion dans laquelle nous situerons clairement la problématique traitée.

1.2 Problématique

Pourquoi s'intéresser à la détection de cibles ?

La détection de cibles est un problème intéressant dont la résolution apporte des avantages et gain de temps couvrant diverses besoins quotidiens qui s'étalent sur divers domaines.

Nous nous intéressons particulièrement à la recherche de mines anti-personnelles et bombes ainsi que la recherche de sources radioactives, car rien n'est plus important que la sécurité et la mise hors de danger de vies humaines.

Pour cela, nous aurons recours à des méthodes de résolution basées sur l'intelligence en essaim. Ces méthodes ont permis dernièrement de maintes avancées, jugées les plus adéquates comme technique de résolution.

De ce fait, nous aurons besoin de développer un système multi-robots où chaque robot se comportera comme une particule de l'essaim.

L'environnement inconnu est considéré en 2D, chaque robot n'a connaissance que du nombre de cibles recherchées, les bordures ou frontières de l'environnement de recherche ainsi que sa position initiale (position avant le début de la recherche). Aussi l'environnement est complexe et englobe plusieurs obstacles, entravant son exploration.

Les positions des obstacles sont inconnues, c'est pourquoi les robots doivent être munis de capteurs adaptés à leur stratégie d'évitement d'obstacles. Les cibles (mines / sources de radiation) dont les positions sont inconnues, émettent un signal que les robots sont capables de recevoir via leurs capteurs.

La validation du système sera effectuée par des expérimentations sur des environnements dont les positions des obstacles, cibles et des robots sont générées aléatoirement. Cela conformément à la procédure présentée dans des travaux récents.

1.3 Paramètres du problème

Plusieurs paramètres entrent en jeu concernant le problème de recherche de cibles, donnant lieu à une arborescence de sous-classes de problèmes (voir figure 1.1), qui se diffèrent par les techniques de résolutions adaptées.

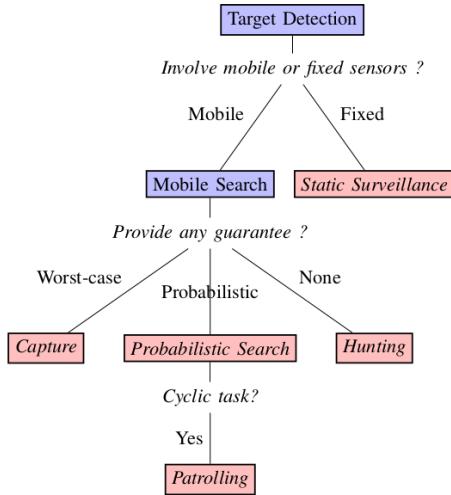


Figure 1.1: Schéma résumant les classes de problèmes de détection de cibles [surv2].

Nous distinguons quatre types de problèmes :

- **La surveillance statique** implique l'utilisation de robots fixes dans un environnement connu afin d'entièrement le couvrir pour y détecter les cibles [surv2].
- **La capture** a pour but de capturer toutes les cibles présentes dans un environnement connu. Seules les positions initiales des cibles ne sont pas connues par le(s) poursuiveur(s) [surv2]. La figure 1.2 illustre un exemple de capture.

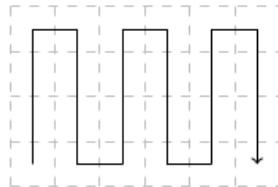


Figure 1.2: Exemple de capture [surv2].

- **La patrouille** a une formulation proche de celle de capture, mais avec un aspect cyclique. Autrement dit la zone n'est pas couverte qu'une seule fois, mais plusieurs fois [surv2], comme le montre la figure 1.3.

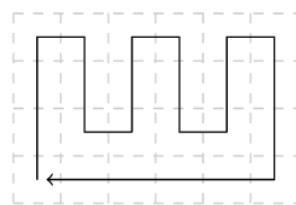


Figure 1.3: Exemple de patrouille [surv2].

- **La chasse** s'attaque à la détection de cibles sans aucune garantie. C'est-à-dire qu'on suppose qu'on ne connaît ni les positions initiales des cibles ni l'environnement de recherche [surv2].

À partir de la formulation de la problématique et des classes de problèmes présentées ci-dessus, on peut clairement situer notre contribution, qui s'inscrit dans le cadre des problèmes de type **Chasse**.

Dans ce qui suit nous dériverons les paramètres du problème les plus pertinents :

1.3.1 Les systèmes multi-robots (SMRs)

1.3.1.1 Définition

Un système multi-robots est un groupe de robots mobiles autonomes ayant un objectif ou un ensemble de tâches communes, ces robots ont la capacité de communiquer et se coordonner afin d'atteindre leur objectif [SMR1].

Les robots mobiles se déplacent durant la recherche selon une stratégie visant à aboutir à des solutions optimales.

Il est important de souligner que le mode de mobilité des robots influence fortement la résolution du problème (vision, vitesse, agilité du mouvement) [surv1].

1.3.1.2 Caractéristiques des SMRs [surv1]

Les systèmes multi-robots se caractérisent par multiples aspects qui font leur force, à commencer par l'exécution des tâches en parallèle, ainsi qu'une couverture d'une plage assez importante de l'espace de recherche. D'autres particularités des SMRs spécifiques aux essaims de robots existent, telles que:

- *La robustesse*, qui offre une tolérance aux pannes.
- *la flexibilité* permet une rapide adaptation aux nouvelles exigences et différents environnements (favorisée par la redondance et la simplicité des comportements des robots).
- *l'évolutivité* qui est la capacité à fonctionner avec un nombre plus grand ou plus petit de robots sans impact considérable sur la performance.

1.3.1.3 Système de communication [surv2]

Deux types de systèmes sont les plus répandus: centralisés et décentralisés, ils sont choisis selon le besoin et les critères recherchés.

- **Les systèmes centralisés**, sont plus aptes à fournir une optimalité globale. Cependant, ils sont souvent confrontés à de fortes contraintes du monde réel, d'où la nécessité d'une connectivité totale dans de nombreuses situations.
- **Les systèmes décentralisés**, quant à eux fournissent des solutions jugées sous-optimales. Par contre, ils sont plus robustes et plus flexibles de par leur adaptation aux environnements dynamiques ce qui leur permet d'aboutir à des performances intéressantes.

1.3.1.4 Comportement des robots [surv2]

Les robots peuvent s'approprier deux types de comportements, soient:

- **Indépendant**, chaque robot est responsable d'un certain nombre de tâches à exécuter.
- **Coopératif**, chaque décision ou action est diffusée à toute l'équipe. Ce mode coopératif peut être explicite, consistant à influencer un robot par une communication directe, ou bien implicite, qui est une prise de décision indépendante selon les informations individuellement recueillies.

1.3.2 Les capteurs

Selon l'environnement à explorer, le besoin en types de capteurs diffère, que ce soit pour reconnaître les cibles, détecter les obstacles afin de les éviter ou reconnaître les autres robots.

Parmi la panoplie des capteurs existants, on trouve : les caméras, les infrarouges, les capteurs lasers (distance), les capteurs à ultrason, la kinect, les capteurs de pression, les capteurs de gaz, GPS, ...etc [capteur] qui sont les plus utilisés. Le choix des capteurs consiste à trouver le meilleur compromis entre coût et efficacité selon les contraintes imposées.

1.3.3 Les cibles

Le nombre de cibles est un paramètre fixé pour un environnement donné. Les problèmes **mono-cible** consistent à trouver la position d'une cible particulière dans l'environnement. En revanche, les problèmes **multi-cibles** visent à maximiser le nombre de cibles trouvées.

Pour le problème de recherche ou détection, les cibles possèdent des positions inchangées dans l'environnement durant toute la recherche [surv1].

De plus, il est toujours supposé que les cibles émettent des radiations, sons, ondes, lumières, odeurs ou autres, qui sont maximales aux alentours de la cible s'estompant graduellement en s'éloignant de celle-ci. L'étendue du champ d'émission de ces cibles est aussi un paramètre qui influe directement l'efficacité des algorithmes de résolution.

1.3.4 L'environnement [surv2]

L'environnement est l'espace de recherche où évoluent les cibles et interviennent les robots comme fouilleurs. Il est décrit comme **connu** lorsqu'une carte est mise à disposition et **inconnu** lorsqu'on manque d'informations globales, n'ayant connaissance que de ses bordures. Ces environnements peuvent être des :

- **Environnements simples**, c'est-à-dire sans obstacles ne contenant aucune contrainte susceptible de bloquer la vue ou la mobilité des robots.
- **Environnements avec obstacles ou complexes**, Ceux-ci contenant des obstacles, contraignants ainsi les robots dans leurs déplacements et leurs perceptions, les poussant parfois à changer de trajectoire.

1.3.5 Les obstacles

Les obstacles sont des contraintes supplémentaires dans le processus de recherche de cibles. Divers types d'obstacles existent, citons les obstacles fixes et mobiles, les obstacles franchissables mais entravant la vue ou encore les obstacles empêchant uniquement le passage sans obstruer la visibilité des robots, ... etc [surv2].

La présence d'obstacles dans les environnements de recherche impose aux méthodes de résolution une prise en considération d'une stratégie d'évitement d'obstacles, pour cela nombreuses sont les stratégies existantes.

Elles s'appuient sur les informations fournies par les capteurs embarqués afin d'orienter les robots à chaque laps de temps pour une réaction en temps réel [Sara].

1.4 Travaux connexes

Plusieurs approches ont été exploitées pour la résolution du problème de détection de cibles dans des environnements inconnus, nous citons à titre d'exemple:

1.4.1 Les algorithmes anti-regroupements [Miao2010]

Souvent appliqués aux problèmes de type "chasse", ces algorithmes imitent le comportement social des animaux solitaires (tigre, araignée, ...). Ils se basent sur trois principales règles, soient:

- La prévention des collisions : en s'éloignant des obstacles les plus proches et en respectant une distance de sécurité.
- La décentralisation : en tentant de se séparer de ses voisins.
- L'égoïsme : si aucune des deux situations précédentes ne se produit, chaque robot choisit la direction permettant de maximiser ses propres gains.

1.4.2 Approche des champs de potentiel [surv2]

Dans cette approche on considère deux vecteurs de forces locales qui influencent les mouvements des robots telles que, les cibles proches émettent des forces attractives vers elles, tandis que les robots proches émettent des forces répulsives éloignant ainsi les autres robots. L'algorithme de Parker pour le contrôle distribué A-CMOMMT est inspiré des champs de potentiels [Parker2002].

1.4.3 Approches basées intelligence en essaim

L'intelligence en essaim et les méta-heuristiques ont été très exploitées pour ce type de problème grâce à la facilité des opérations stochastiques qui les distinguent des autres techniques. Leur force majeure est l'inspiration des comportements d'espèces ou phénomènes naturels [robotMeta].

PSO, GA, ACO, BA, ABC, BFO, GSO et FA sont les techniques de méta-heuristiques les plus utilisées revenant souvent dans la littérature avec des améliorations et contributions apportées par les auteurs [surv1].

Quant aux travaux récents on trouve :

1.4.3.1 BSO (Bee Swarm Optimization) (2013)

Une variante de l'algorithme Bee Swarm Optimization inspirée du comportement des abeilles dont les auteurs sont Hannaneh Najd Ataei, Koorush Ziarati et Mohammad Eghatesad [BSO2013] comporte trois types d'abeilles (abeilles butineuses, abeilles éclaireuses, abeilles spectatrices) chacune possède sa propre fonction de mise à jour de ses positions.

Cette variante a montré de meilleures performances que la version classique de PSO. Suite à des expérimentations sur un ensemble d'environnements aléatoires, il a été prouvé que cette approche de BSO peut être 50,6% plus efficace que PSO.

Mais elle n'a été testée que sur des environnements mono-cibles. De plus, la fonction objectif utilisée s'appuie sur la distance euclidienne entre les robots et la cible, or que les positions des cibles sont supposées inconnues.

1.4.3.2 A-RPSO

A-RPSO (Adaptative Robotic Particule Swarm Optimization) est une technique basée essaim proposée par les auteurs M. Dadgar, et al., où chaque particule représente un robot. La contribution de l'auteur par rapport à un PSO classique est l'ajout d'une stratégie d'évitement d'obstacles incluse dans l'équation de déplacement, ainsi qu'une mise à jour des paramètres classiquement constants comme le poids d'inertie de façon dynamique, dite adaptative.[**Dadgar2016**].

Ces modifications ont eu comme objectif d'éviter les optimaux locaux, mieux contrôler et adapter l'allure des robots lors du rapprochement des cibles et des obstacles. La technique a été testée sur un simulateur 2D créé par les auteurs, elle reste proche d'une modélisation d'une grille virtuelle bidimensionnelle.

1.4.3.3 MFPSO (Multi-swarm hybrid FOA-PSO)

Un nouvel algorithme multi-essaim hybride (MFPSO) basé sur FOA (Fruit Fly Optimization Algorithm) et PSO (Particule Swarm Optimization) pour la recherche d'une cible dans des environnements inconnus a été présenté par les auteurs, Hongwei Tang, Wei Sun, Hongshan Yu, Anping Lin, Min Xue et Yuxue Song [**novel**]. Cette nouvelle mét-heuristique se singularise par les avantages suivants :

- Parallélisme grâce à un coefficient adaptatif.
- Présence de stratégies contournant la convergence prématuée grâce à l'indépendance des essaims de robots.
- Mécanisme d'évacuation (MSCM) comme stratégie d'évitement d'obstacles.

Les nombreuses expériences effectuées pour le mode mono-cible sur MFPSO prouvent qu'il surpasse de manière significative les autres approches (dont A-RPSO et RPSO), en termes de nombre d'itérations et taux de réussite, dans la plupart des cas.

Par contre, le champ d'émission de la cible est inconnu, or comme mentionné plus haut, ce paramètre influence la difficulté du problème.

Les tests effectués comportent le cas du mono-cible seulement, ce qui implique que le comportement et l'efficacité de cet algorithme dans des environnements multi-cibles sont inconnus.

Aussi, le champ de vue des robots a été fixé par les auteurs à une valeur de dix pixels durant toutes les expérimentations.

Enfin, les auteurs utilisent une fonction objectif qui manipule la distance $Dist(x_i^t)$ entre les robots et la cible. Cependant, la position de la cible est supposée inconnue. C'est pourquoi cette distance au lieu d'être directement utilisée comme fonction objectif elle a servi à calculer une estimation des signaux émis par la cible.

1.5 Conclusion

Ce chapitre nous a permis de décortiquer les différentes variantes du problème de détection de cibles ce qui nous a permis de déduire que notre travail visera à proposer une solution au problème de *Chasse*.

Tous les paramètres nécessaires pour clairement définir notre problème et aboutir à une bonne modélisation, ont été explicitement exposés. Notre méthode de résolution est basée intelligence en essaim, d'où la nécessité d'éclaircir certains aspects incontournables relatifs aux approches choisies dans le chapitre suivant.

Chapitre 2 : Intelligence en essaim : BSO, EHO, EWSA et GA

2.1 Introduction

Dans ce chapitre nous allons présenter les outils intelligents de résolution pour notre problématique. Ces derniers consistent en des méta-heuristiques inspirées de l'intelligence collective des animaux et insectes, autrement dit l'intelligence en essaim.

Nous commencerons par introduire les algorithmes basés intelligence en essaim et leurs principaux paramètres, suivis d'un décryptage des trois approches choisies pour notre étude, à savoir :

- Bee Swarm Optimization (BSO) qui est un algorithme de résolution s'inspirant de l'intelligence collective du comportement des abeilles lors de la recherche de nourriture. BSO a fait ses preuves face à de nombreux problèmes d'optimisation combinatoire (le problème de satisfiabilité SAT, le problème du voyageur de commerce, ...etc.).
- Elephant Herding Optimisation (EHO) et Elephant Swarm Water Search Algorithm (EWSA), tous deux des algorithmes inspirés du comportement des éléphants en groupe.
- On verra aussi l'algorithme génétique (GA) dans sa version GA incrémental.

On clôturera ce chapitre par les motivations qui nous ont poussées à choisir ces algorithmes intelligents comme méthodes de résolution pour la détection de cibles.

2.2 Les algorithmes basées intelligence en essaim

Swarm Intelligence ou l'intelligence en essaim étudie le comportement coopératif entre les espèces naturelles. C'est un domaine prometteur de l'intelligence artificielle dont le principe de base repose sur l'observation par la nature des phénomènes intelligents et surtout des comportements de groupe chez les animaux.

Une approche ou méthode basée intelligence en essaim est également considérée comme une méta-heuristique. Elle suit un schéma d'algorithme évolutif, dicté par la simulation de l'évolution des espèces naturelles [[courshabiba](#)].

Les méta-heuristiques sont des approches génériques qui visent à proposer des solutions à des problèmes complexes, en employant une certaine stratégie d'exploration de l'espace de recherche appelée **espace des solutions**. Ce dernier comporte toutes les solutions possibles au problème. Ces **solutions** sont ensuite filtrées grâce à une **fonction objectif** permettant de les ordonner selon leur optimalité [[courshabiba](#)].

La figure 2.1 montre qu'il existe deux catégories principales de méta-heuristiques. Les méta-heuristiques basées sur une solution unique, telles que, la recherche tabou (TS) [[aouatTS](#)], recuit simulé, la recherche locale guidée (GLS) [[habibaGLS](#)], ...etc, et les méta-heuristiques basées population comme l'algorithme génétique (GA) [[kechidGA](#)] ou une de ses extensions, Les algorithmes mémétiques (MA) [[habibaMA](#)], la recherche par dispersion (SS) [[habibaSS-GA](#)], l'algorithme Bat (BA) [[boukraBA-ACO](#), [habibaBA2](#)], les colonies de fourmis (ACS) [[habibaACO](#)], l'algorithme d'optimisation par essaim de particules (PSO) [[habibaPSO](#)], l'algorithme des lucioles FFA (Firefly algorithm) [[habibaFAA](#)],

l'essaim d'abeilles à savoir BSO (Bee Swarm Optimization) [habibaBSO1, kechidBSO], ABSO (Advanced Bee Swarm Optimization) [habibaABSO], ou encore ABC (Artificial Bee Colony) [boukraABC] etc.

Les algorithmes basés intelligence en essaim peuvent être vus comme des méta-heuristiques basées population [courshabiba].

Ces nombreux travaux ont été développés au sein du laboratoire de recherche en l'intelligence artificielle LRIA¹.

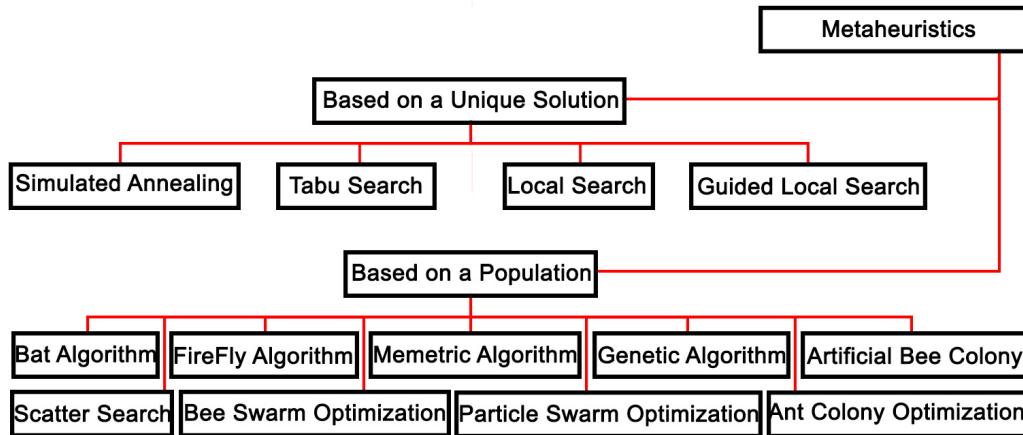


Figure 2.1: Classification des méta-heuristiques

Étant donné l'efficacité et la simplicité des méta-heuristiques, on va s'intéresser de plus près à BSO, EHO, EWSA et GA qui sont des méta-heuristiques basées population ayant connu un large intérêt et succès.

2.3 Bee Swarm Optimization [Drias]

2.3.1 Inspiration des abeilles

De nombreuses études biologiques se sont intéressées aux comportements et à la psychologie des abeilles afin de décortiquer leurs méthodologies et stratégies de recherche du nectar ainsi que leur façon de communiquer les informations concernant leurs trouvailles.

L'une des expériences les plus connues sur les abeilles, est celle de Seeley, Camazine et Sneyd en 1991 [Seeley], elle a abouti sur les quelques caractéristiques suivantes :

- la catégorie d'abeilles "*travailleuses*" est celle qui se charge de trouver la nourriture ("Scout bee"). Les abeilles reviennent à la ruche en ramenant un peu de leurs récoltes.
- Les abeilles ont tendance à privilégier les régions qui regorgent de nourriture aussi lointaines soient elles aux zones moins riches en pollen, mais à proximité de la ruche.
- L'abeille ayant trouvé la meilleure région en termes de nourriture effectuera la danse la plus rigoureuse, ce qui guidera l'essaim vers la meilleure région.
- La danse (mouvement circulaire) est le moyen de communication entre les abeilles, elle englobe les informations concernant la quantité, la direction et la distance entre la ruche et la région où se trouve la nourriture.

¹<https://www.lria.usthb.dz/>

2.3.2 Analogie entre le comportement des abeilles et BSO

BSO est entièrement inspirée du comportement des abeilles pour la recherche du pollen, ci-dessous le tableau 2.1 fait l'analogie entre le comportement des abeilles travailleuses et leurs représentations simplifiées dans BSO :

Essaim d'abeilles	BSO
Une abeille éclaireuse est envoyée dans un champ de fleurs et se positionne sur une fleur aléatoirement.	Une solution est générée aléatoirement à partir de l'espace des solutions, elle est ensuite affectée à l'abeille de référence.
Les abeilles s'orientent vers les champs de fleurs à la recherche de quantité de nourriture (pollen).	Les abeilles artificielles se déplacent dans l'espace des solutions à la recherche de bonnes solutions.
Les abeilles se dispersent autour de la zone indiquée par l'abeille éclaireuse pour couvrir au mieux le champ de fleurs.	La diversification est basée sur le Flip. Ce dernier permet la génération d'un certain nombre de solutions à partir de la solution de référence.
Chaque abeille procède à une recherche locale dans sa zone d'atterrissement.	Chaque abeille porteuse d'une solution effectue une recherche locale, ce processus est appelé " <i>intensification</i> ".
Chaque zone de fleurs est évaluée selon la quantité de pollen qui s'y trouve.	Chaque solution est évaluée selon la fonction objectif.
Chaque abeille se rappelle de la fleur source ayant la plus grande quantité de pollen ainsi que l'endroit où elle se trouve. Elle se met à danser juste au-dessus pour en informer l'essaim.	Chaque abeille sauvegarde sa meilleure solution locale dans une table appelée <i>Table Dance</i> .
La vigueur de la danse de l'abeille est proportionnelle à la richesse et quantité de nourriture trouvée.	La sélection d'une solution de la table <i>Dance</i> se base sur un critère de qualité. Toutes les abeilles sont mises au courant de la meilleure solution actuelle.
En cas de recherche non-fructueuse, les abeilles explorent d'autres zones distantes	en cas de stagnation de la recherche, la solution la plus diverse de la table <i>Dance</i> est sélectionnée comme solution de référence.

Table 2.1: Analogie entre les caractéristiques des essaims d'abeilles et BSO.

2.3.3 Paramètres et fonctions

BSO est caractérisée par plusieurs paramètres empiriques, qui sont :

- **nbrBee** : le nombre d'abeilles de l'essaim.
- **Flip** : la fréquence d'inversement.
- **MaxChance** : le nombre de chances maximal.
- **MaxItération** : le nombre d'itérations maximal.

Sref: La solution de référence est initialisée aléatoirement au début de la recherche, puis mise à jour selon les étapes de BSO, par la meilleure solution en termes de qualité ou de diversité.

Flip: C'est un paramètre qui détermine le nombre de variables à inverser au niveau de la solution de référence (Sref) afin d'obtenir de nouvelles solutions. $\frac{n}{flip}$ représente ainsi la distance qui sépare les nouvelles solutions de *Sref*, *n* étant la taille de la solution.

Détermination des zones de recherche: l'application du *Flip* sur une solution de référence génère un ensemble de solutions équidistantes de *Sref*, formant un cercle autour d'elle.

Recherche locale: Chaque abeille effectue une recherche locale à partir de la solution qui lui est affectée afin d'évaluer les solutions voisines en ne gardant que la meilleure.

MaxChance: Chaque solution possède un nombre de chances maximal pour être choisie comme solution de référence à chaque itération.

Table dance: La table "*Dance*" est le répertoire commun à toutes les abeilles de l'essaim pour y inscrire leurs meilleures solutions locales.

2.3.4 Pseudo-code BSO

```

1 Algorithme : BSO
2
3   Input : MaxIteration, Flip, Maxchances, Optimale, nombred'abeilles
4   Output : meilleureSolution : la meilleure solution trouvée
5   /* génération aléatoire de la solution de référence. */ 
6   Sref ← solutionAléatoire()
7   meilleureSolution ← Sref
8   /* Tant que le nombre d'itérations maximal n'est pas atteint et la
    solution optimale n'est pas trouvée réitérer */ 
9   while ¬(MaxIteration or Optimale) do
10    /* Insertion de la solution de référence dans la liste Tabou */
11    Insertion (listeTabou,Sref)
12    /* Affectation des zones de recherche aux abeilles */
13    abeilles ← DeterminationDesZonesDeRecherche (Sref)
14    foreach abeille ∈ abeilles do
15      /* Effectuation d'une recherche locale */
16      solutionLocale ← rechercheLocale (abeille)
17      /* Evaluation et insertion de la meilleure solution locale dans
       la table Dance. */
18      Insertion (Dance,MeilleuresolutionLocale)
19    end
20    /* Sélection de la meilleure solution de la table Dance. */
21    Sref ← MeilleureDeDance (Dance)
22    if sRef > meilleure then
23      /* Sauvegarde de la nouvelle meilleure solution. */
24      meilleureSolution ← sRef
25    end
26  end
27  return meilleureSolution

```

Algorithme 1 : BSO Générique

```

1 Fonction : DéterminationDesZonesDeRecherche
2
3   Input : Flip : fréquence d'inversement
4   Output : ensemble_K_solutions : ensemble de K solutions
5   h ← 0
6   while taille(espace_recherche) < k and h < Flip do
7     s ← Sref; p ← 0
8     /* Inversion partielle de la solution, à une fréquence : flip */
9     repeat
10      s ← Inverser (s,s[Flip * p + h])
11      p ← p + 1
12    until Flip * p + h > n
13    /* Insertion de la solution générée dans la liste de solutions */
14    ensemble_K_solutions ← ensemble_K_solution ∪ {s}
15    h ← h + 1
16  end
17  return ensemble_K_solutions

```

Fonction DéterminationDesZonesDeRecherche

2.4 EHO & EWSA : Deux algorithmes basés essaim d'éléphants

2.4.1 Inspiration des éléphants

Les éléphants sont des mammifères sociaux qui présentent des structures sociales complexes ayant une multitude de particularités surprenantes.

Dans un premier temps, on se contentera d'exhiber les caractéristiques individuelles suivantes [EHO1] :

- Bonne mémoire.
- Font preuve d'une intelligence avancée (reconnaissance de soi, la conscience de soi).
- Capacité d'apprendre et de distinguer plusieurs discriminations visuelles et acoustiques.
- Capacité à utiliser et manipuler des outils de la vie réelle.
- Possession de systèmes de détection et de communication très avancés.

Quant aux caractéristiques de groupe, nous citons [EHO2] :

- Un groupe d'éléphants est composé de plusieurs clans.
- Un clan est dirigé par une matriarche (souvent l'éléphant le plus vieux du troupeau).
- Les femelles préfèrent vivre dans des groupes familiaux (clan).
- Les éléphants mâles ont tendance à quitter leur groupe familial en grandissant.
- Les éléphants mâles loin de leur groupe familial peuvent rester en contact avec les éléphants de leur clan par le biais de vibrations basses fréquences.

Méthodes de communication [EHO1]:

Les éléphants utilisent leurs sens de l'ouïe, de l'odorat, de la vision, du toucher et une capacité exceptionnelle à détecter les vibrations pour communiquer entre eux ainsi qu'avec les autres espèces. Il existe deux types courants de communication entre éléphants:

1. Communication à longue distance: (jusqu'à 10-12 km)

-*Les communications sismiques*, causées par le grondement et le mouvement des éléphants. Leur réception se fait à travers leurs mécano-récepteurs dans les orteils ou les pieds et la pointe de leurs trompes.

-*Les communications sonores* produisent une gamme de signaux sonores (infrasons) que leurs trompes peuvent amplifier. Leurs oreilles font office de parabole pour la réception des sons de basses fréquences d'autres éléphants lointains.

-*Les communications chimiques (olfactives)*, ils utilisent leurs trompes pour sentir l'air, ou pour explorer le sol et les arbres, ainsi que pour renifler d'autres éléphants.

2. Communication à courte distance:

-*Les communications visuelles*, la vision des éléphants atteint une portée maximale de 46-100 m. Ils utilisent la tête, les yeux, la bouche, les oreilles, la trompe et tout le corps pour l'échange de messages.

-*Les communications tactiles*, les éléphants sont des animaux extrêmement tactiles. Ils communiquent par le toucher en utilisant leurs trompes, leurs défenses, leurs pieds, leurs queues, ...etc.

2.4.2 EHO : Elephant Herding Optimization

2.4.2.1 Analogie entre les éléphants et EHO [EHO2]

La météo-heuristique **EHO** simule l'évolution du mouvement des éléphants. Elle représente un comportement simplifié des groupes d'éléphants se basant sur les règles idéalisées présentées dans le tableau qui suit :

Clans d'éléphants	EHO
Les éléphants parcouruent la nature à la recherche de meilleures sources de nourriture et d'eau	Les éléphants artificiels se déplacent dans l'espace des solutions cherchant la meilleure solution globale au problème à traiter.
La quantité de nourriture et d'eau permet aux éléphants d'évaluer la qualité de l'environnement	La qualité de la solution est évaluée à travers la fonction objectif.
Les éléphants vivent en clans où chaque clan est composé de plusieurs éléphants.	Plusieurs clans (nClan) sont créés avec un nombre fixe d'éléphants N , une solution est affectée à chaque éléphant.
Les éléphants de chaque clan vivent sous la direction d'une <i>matriarche</i> qui guide leurs déplacements.	L'éléphant possédant la meilleure solution du clan (meilleure locale) guide le clan.
Un nombre déterminé d'éléphants mâles quittera leur groupe familial et vivra solitairement loin du groupe d'éléphants principal à chaque génération.	Le pire éléphant en termes d'évaluation, devra se détacher de son clan et explorer une nouvelle zone à chaque itération.

Table 2.2: Analogie entre les caractéristiques des éléphants et EHO.

2.4.2.2 Paramètres et fonctions [EHO2]

Cette méthode repose sur un certain nombre de paramètres empiriques, qui sont :

- **nClan** : le nombre de clans.
- **N** : le nombre d'éléphants d'un clan.
- α : le taux d'influence de la matriarche sur un éléphant.
- β : le taux d'influence du centre de gravité du clan sur la matriarche.

Pour le calcul des positions futures des éléphants et de la dynamique des clans, l'algorithme est basé sur trois fonctions principales, soient :

- Pour chaque éléphant **i** du clan **c**, sa prochaine position est calculée par l'équation suivante :

$$X_{new,c,i} = X_{c,i} + \alpha * (X_{Best,c} - X_{c,i}) * r \quad (2.1)$$

Avec: $r \in [0, 1]$: distribution uniforme.

$X_{c,i}$ et $X_{new,c,i}$: positions (avant et après) du $i^{\text{ème}}$ éléphant dans le clan c.

$\alpha \in [0,1]$: taux d'influence de la matriarche sur le $i^{\text{ème}}$ éléphant.

- Pour l'éléphant matriarche du clan c, sa position future est calculée comme suit:

$$X_{Best,c} = \beta * X_{center,c} \quad (2.2)$$

Avec: $X_{center,c}$: le centre de gravité du clan c.

β : le taux d'influence du centre de gravité du clan sur la matriarche.

Sachant que le centre de gravité est donné par la formule suivante:

$$X_{center,c} = \frac{1}{N} * \sum_{i=1}^N X_{c,i} \quad (2.3)$$

- Pour le pire éléphant (*Worst*) qui quittera le clan, sa position est déterminée par l'estimation:

$$X_{Worst,c} = X_{min} + (X_{max} - X_{min}) * rand \quad (2.4)$$

Avec: $rand \in [0, 1]$: distribution stochastique uniforme.

X_{min} et X_{max} : limites (inférieure, supérieure) de la position d'un éléphant.

2.4.2.3 Pseudo code de EHO [EHO2]

```

1 Algorithme : EHO
2
3   Input : MaxGen : le nombre maximum de générations,
4   Output : meilleure solution
5   begin
6     Initialisation
7     t=1; /* Initialisation du compteur de générations */ 
8     initPopulation();/* initialisation des solutions des éléphants. */
9     while t < MaxGen do
10    /* Tri de tous les éléphants selon la fonction d'évaluation.
11       */
12    TriElephant ();
13    /* Mise à jour de la position des éléphants de chaque clan. */
14    MAJPositionClan ();
15    /* Mise à jour de la position du pire éléphant de chaque clan.
16       */
17    OperateurSéparation ();
18    /* Évaluation de la population. */
19    EvaluationPopulation ();
20    t ← t + 1;
21   end
22 end
```

Algorithme 2 : EHO Générique

```

1 Fonction : MAJPositionClan
2
3   Input : nClan : nombre de clans, N : nombre d'éléphants d'un clan
4   /* pour chaque clan de la population */ *
5   for c ← 1 to nClan do
6     /* pour chaque éléphant du clan c */ *
7     for i ← 1 to N do
8       Mise à jour de  $X_{c,i}$  et calcul de  $X_{new,c,i}$  selon Eq. (2.1).
9       if  $X_{c,i} = X_{Best,c}$  then
10         | Mise à jour de  $X_{c,i}$  et calcul de  $X_{new,c,i}$  selon Eq. (2.2).
11       end
12     end
13   end
14 
```

Fonction MAJPositionClan

```

1 Fonction : OpérationSéparation
2
3   Input : nClan : nombre de clan de la population
4   /* pour chaque clan de la population */ *
5   for c ← 1 to nClan do
6     | Remplacer le pire éléphant du clan c selon Eq. (2.4).
7   end
8 
```

Fonction OpérationSéparation

2.4.3 EWSA : Elephant Swarm Water Search Algorithm

2.4.3.1 Analogie entre les éléphants et EWSA [EHO1]

L'algorithme évolutionnaire EWSA proposé par *S Mandal*, simule aussi l'évolution du mouvement des éléphants mais pour la recherche d'eau. La métaphore entre le milieu naturel et la résolution de problèmes est présentée dans le tableau 2.3 suivant :

Groupes d'éléphants	EWSA
Les éléphants sont à la recherche de meilleures sources d'eau	Les éléphants cherchent la meilleure solution de l'espace des solutions.
L'endroit où se trouve un éléphant dans son territoire.	La position d'une solution dans l'espace de recherche.
Le déplacement d'un éléphant se fait selon sa vitesse.	Le changement de position d'une solution dans l'espace de recherche est calculé sur la base de la vitesse de l'éléphant artificiel.
Un meilleur niveau d'eau dénote une meilleure zone pour la survie des éléphants.	Une meilleure solution est déterminée par une meilleure évaluation par la fonction objectif.
Les éléphants forment plusieurs groupes, formant un essaim d'éléphants. Chaque groupe est composé d'un certain nombre d'éléphants	Les groupes (ou clans) d'éléphants sont représentés par un éléphant unique nommé chef du clan.
Capacité de communication à grande distance et à courte distance des éléphants.	Recherche globale et locale des éléphants.

Chaque fois qu'un groupe d'éléphants trouve une source d'eau, il communique aux autres groupes de l'essaim la quantité d'eau trouvée.	les groupes d'éléphants communiquent entre eux les solutions trouvées et leur évaluation.
Les groupes d'éléphants à la recherche d'une source d'eau, gardent en mémoire la meilleure source trouvée en plus de celle trouvée par d'autres groupes communiquant avec eux	Chaque groupe d'éléphants se souvient de sa propre meilleure solution (locale), et de la meilleure solution de tout l'essaim (globale).
La proximité physique entre groupes d'éléphants et d'autres facteurs tels que l'atténuation du signal à grande distance influent la recherche d'eau.	La recherche de solution locales et globales est contrôlée par une constante probabiliste p .

Table 2.3: Analogie entre les caractéristiques des éléphants et EWSA.

2.4.3.2 Paramètres et fonctions [EHO1]

L'ensemble des paramètres empiriques de cette méthode sont :

- N : le nombre de groupes.
- p : la constante de commutation (pour basculer entre la recherche globale et locale).
- w^t : le poids d'inertie à l'itération t (pour équilibrer entre exploration et exploitation).

Afin de calculer la position et vitesse suivantes des éléphants, l'algorithme repose sur un ensemble de fonctions. Mais avant, nous devons définir le format dans lequel sont représentées les positions et vitesses des groupes d'éléphants, nous avons:

$V_{i,d}^t = (v_{i1}, v_{i2}, \dots, v_{id})$: la vitesse du i^{eme} groupe d'éléphant à l'itération t .

$X_{i,d}^t = (x_{i1}, x_{i2}, \dots, x_{id})$: la position du i^{eme} groupe d'éléphant à l'itération t .

$P_{Best,i,d}^t = (p_{i1}, p_{i2}, \dots, p_{id})$: la meilleure position du i^{eme} groupe d'éléphant à l'itération t .

$G_{Best,d}^t = (g_1, g_2, \dots, g_d)$: la meilleure position de tous l'essaim d'éléphant à l'itération t .

X_{max} et X_{min} : limite supérieure et limite inférieure des positions (dans l'environnement).

d : la dimension d'une solution.

- Pour chaque groupe d'éléphants i , sa vitesse est donnée par la formule:

$$V_{i,d}^{t+1} = \begin{cases} V_{i,d}^t * w^t + rand(1, d) \odot (G_{best,d}^t - X_{i,d}^t) & \text{si } random > p \\ V_{i,d}^t * w^t + rand(1, d) \odot (P_{best,i,d}^t - X_{i,d}^t) & \text{si } random \leq p \end{cases} \quad (2.5)$$

Ainsi on obtient sa position par l'équation:

$$X_{i,d}^{t+1} = V_{i,d}^{t+1} + X_{i,d}^t \quad (2.6)$$

Avec:

$rand(1, d) \in [0,1]$: tableau de dimension d de valeurs aléatoires.

w^t : le poids d'inertie à l'itération t .

$random \in [0,1]$: un chiffre aléatoire entre $[0,1]$.

p : constante de commutation.
 \odot : multiplication élément par élément.

- Le poids d'inertie à l'itération t , w^t peut être calculé de diverses manières dont les plus utilisées par S.Mandal sont :

- Poids d'inertie constant / Constant Inertia Weight (CIW):

$$w^t = \text{constante} \quad (\text{généralement} = 0.5) \quad (2.7)$$

- Poids d'inertie aléatoire / Random Inertia Weight (RIW):

$$w^t = 0.5 + \text{rand}/2 \quad (\text{rand} \in [0, 1]) \quad (2.8)$$

- Poids d'inertie décroissant linéairement/ Linearly Decreasing Inertia Weight(LDIW):

$$w^t = w_{\max} - \frac{w_{\max} - w_{\min}}{t_{\max}} * t \quad (2.9)$$

Avec:

w_{\max} et w_{\min} : la valeur maximale et minimale d'inertie.

t : le numéro de l'itération en court.

t_{\max} : le nombre maximal d'itérations.

2.4.3.3 Pseudo code de EWSA [EHO1]

```

1 Algorithme : EWSA
2
3   Input :  $N$  : nombre de groupes d'éléphants,  $p$  : constante de commutation,
4      $X_{min}$  : borne minimale d'une solution,  $X_{max}$  : borne maximale d'une
5     solution,
6      $t_{max}$  : nombre maximal d'itérations
7   Output :  $G_{Best,d}, f(G_{Best,d})$  : meilleure solution et évaluation.
8
9   begin
10    | for  $i \leftarrow 1$  to  $N$  do
11      |   /* Initialisation de la position et vitesse du  $i^{ème}$  groupe. */
12      |   Initialisation  $X_{i,d}$  et  $V_{i,d}$ ;
13      |   /* Initialisation du PBest du  $i^{ème}$  groupe. */
14      |    $P_{Best,i,d} = X_{i,d}$ ;
15      |   /* Évaluation de la position du  $i^{ème}$  groupe. */
16      |   Evaluation  $f(X_{i,d})$ ;
17    | end
18    |  $G_{best,d} = \text{Min}(f)$ ; /* Initialisation du GBest. */
19    | Initialisation  $w^t$  selon Eq. (2.9)
20    | /* Tant que le nombre d'itérations maximal n'est pas atteint */
21    | for  $t \leftarrow 1$  to  $t_{max}$  do
22      |   /* Pour chaque groupe d'éléphants */
23      |   for  $i \leftarrow 1$  to  $N$  do
24        |       if  $random > p$  then
25          |           /* Mise à jour de la vitesse selon le GBest */
26          |           Mise à jour de la vitesse  $V_{i,d}$  1ère partie de l' Eq. (2.5)
27        |       end
28        |       else
29          |           /* Mise à jour de la vitesse selon le PBest (locale) */
30          |           Mise à jour de la vitesse  $V_{i,d}$  2ème partie de l' Eq. (2.5)
31        |       end
32        |       /* Mise à jour de la position du  $i^{ème}$  groupe. */
33        |       Mise à jour de la position  $X_{i,d}$  avec Eq. (2.6);
34        |       /* Évaluation de la position du  $i^{ème}$  groupe. */
35        |       Evaluation  $f(X_{i,d})$ ;
36        |       if  $f(X_{i,d}) < f(P_{Best,i,d}^t)$  then
37          |           |  $P_{Best,i,d}^t = X_{i,d}$  /* Mise à jour du PBest du  $i^{ème}$  groupe. */
38        |       end
39        |       if  $f(P_{Best,i,d}^t) < f(G_{Best,d}^t)$  then
40          |           |  $G_{Best,d}^t = P_{Best,i,d}^t$  /* Mise à jour du GBest */
41        |       end
42    |   end
43  | end
44  | Retourner ( $G_{Best,d}^t, f(G_{Best,d}^t)$ )
45
```

Algorithme 3 : EWSA Générique

2.4.4 Remarque

L'ossature des deux algorithmes EHO et EWSA est différente. EHO, la version classique prend en compte la composition d'un clan de plusieurs éléphants contrairement à EWSA qui réduit chaque clan en une seule unité.

Aussi, EHO possède deux facteurs de diversification qui sont le multi-clans et l'opérateur de séparation, mais une intensification ne menant pas toujours à une convergence. À l'opposé de l'approche EWSA qui fournit un certain équilibre entre diversification et intensification malgré les risques de convergence prématuress en vue de sa représentation des clans.

Ces deux méta-heuristiques ont prouvé leur efficacité, suite à nombreuses expérimentations et comparaisons avec d'autres approches comme par exemple : CS (Cuckoo Search) [**metaCS**], BA (Bat Algorithm) [**metaBat**] , FPA (Flower Pollination Algorithm) [**FPA**] dans le cas de EWSA. Et BBO (Biogeography-Based Optimization) [**metaBBO**], DE (Differential Evolution) [**metaDE**] et GA (Genetic Algorithm) [**metaGA**] pour EHO.

2.5 Algorithme Génétique [GAthexe]

La première description du processus des algorithmes génétiques a été donnée par Holland en 1975. Puis Goldberg en 1989 les a utilisés pour résoudre des problèmes concrets d'optimisation [**GAthexe**].

2.5.1 Inspiration de la génétique

Les algorithmes génétiques sont des algorithmes évolutionnaires se caractérisant par leur inspiration de l'évolution naturelle des espèces, regroupant les principales composantes de l'ADN, telles que :

- **Chromosome**, c'est une suite d'informations structurées dont l'ordre est important.
- **Gène**, il s'agit d'une unité d'un chromosome, celle-ci renferme une information appelée *Allèle*.
- **Allèle** est une information possédant une position bien définie dans le chromosome. Il s'agit de la valeur que peut prendre un gène et elle est propre à chaque individu.

2.5.2 Métaphore entre l'ADN et un algorithme génétique

L'algorithme génétique a emprunté grand nombre de concepts propres à l'ADN en particulier et à l'évolution des espèces en général, citons les similitudes dans le tableau suivant:

Nature	Algorithme génétique
Une société d'individus.	L'espace de recherche composé de toutes les solutions potentielles.
Une population d'individus appartenant à la société.	Un ensemble de solutions pris de l'espace de recherche.
Le but est la sélection des meilleurs gènes pour former le meilleur individu.	Le but est de trouver la meilleure solution de l'espace des solutions.
Un chromosome permet d'identifier un individu et ses propriétés.	Une solution représente un individu de la population.
Un chromosome regroupe plusieurs informations génétiques	Une solution est formée d'une suite de valeurs.

La reproduction entre deux individus donne naissance à des enfants.	L'opération de croisement entre deux solutions permet d'injecter de nouvelles solutions dans la population.
Pendant la création de l'ADN fils quelques modifications de gènes ou anomalies peuvent survenir.	Après la création de la solution fils, elle peut subir une mutation.
L'accouplement de parents ayant les meilleurs caractères héréditaires est plus susceptible de donner naissance à un fils doté des meilleurs gènes	La sélection des meilleurs parents à chaque itération augmente les chances d'atteindre la solution optimale.

Table 2.4: Analogie entre les caractéristiques de l'ADN et GA.

2.5.3 Paramètres et fonctions

2.5.3.1 Opération de croisement

Le croisement consiste à fragmenter deux solutions qu'on nomme *parents* pour ensuite les regrouper différemment ce qui donne naissance à deux *solutions fils*.

Un point de croisement Généralement, ce point est pris aléatoirement, divisant chaque solution en deux fragments, les deux nouveaux individus prendront chacun une partie des gènes de chaque parent. Le processus est illustré dans la figure 2.2.



Figure 2.2: Croisement en un point.

Deux points de croisement Pour deux points de croisement, chaque parent (solution) devra être fragmentée en trois, ainsi chaque solution fils héritera de trois fragments (1 d'un parent et 2 de l'autre), comme représentés dans la figure 2.3 ci-dessous.



Figure 2.3: Croisement en deux points.

2.5.3.2 Opération de mutation

La mutation apporte l'aléa nécessaire à une exploration efficace de l'espace des solutions en permettant de quitter les extrêmes locaux. Il existe plusieurs façons de procéder à une mutation:

Inversion d'un gène Cette méthode consiste en l'inversement ou la troncature d'un certain nombre fixe ou aléatoire de gènes d'une solution. La figure 2.4 en est un exemple.



Figure 2.4: Mutation avec inversement.

Permutation de gènes Comme le montre la figure 2.5, la permutation de gènes nécessite d'abord la sélection de deux gènes sur l'individu (chromosome) à muter. Puis vient l'échange des deux valeurs (allèles) correspondantes, cela produit un nouvel individu.



Figure 2.5: Mutation avec permutation.

2.5.4 Pseudo code (GA incrémental)

```

1 Algorithme : GA
2
3   Input :  $N$  : taille de la population,  $MaxGen$  : nombre maximal de générations
4   Output : meilleure solution
5   begin
6      $K \leftarrow 0;$ 
7      $P_k \leftarrow \text{GénérationSolutions}(N);$ 
8     Evaluation( $P_k$ ); /* Évaluation de tous les individus de la
9       population */
```

$$\begin{aligned} &10 \quad \text{while } \neg \text{BonneSolution} \& k < MaxGen \text{ do} \\ &11 \quad \quad /* Sélection des deux meilleurs parents de la population */ \\ &12 \quad \quad (Parent1, Parent2) \leftarrow \text{MeilleurIndividu}(P_K); \\ &13 \quad \quad /* Calcul de la position du point de croisement. */ \\ &14 \quad \quad i \leftarrow \text{PointCroisement}(\text{taille}(solution)); \\ &15 \quad \quad /* Croisement des solutions parents */ \\ &16 \quad \quad Fils1 \leftarrow \text{Crosisement}(Parent1, Parent2, i); \\ &17 \quad \quad Fils2 \leftarrow \text{Crosisement}(Parent2, Parent1, i); \\ &18 \quad \quad /* Évaluation des solutions fils */ \\ &19 \quad \quad Evaluation(Fils1); Evaluation(Fils2); \\ &20 \quad \quad /* Détermination du nombre de mutations. */ \\ &21 \quad \quad j \leftarrow \text{NombreMutation}(\text{taille}(solution)); \\ &22 \quad \quad /* Mutation des solutions fils */ \\ &23 \quad \quad FilsMutant1 \leftarrow \text{Mutation}(Fils1, j); \\ &24 \quad \quad FilsMutant2 \leftarrow \text{Mutation}(Fils2, j); \\ &25 \quad \quad /* Évaluation des solutions fils mutantes */ \\ &26 \quad \quad Evaluation(FilsMutant1); Evaluation(FilsMutant2); \\ &27 \quad \quad K \leftarrow K + 1; \\ &28 \quad \quad /* Mise à jour de la population, suppression des solutions
 parents et insertion des solutions fils et fils mutantes. */ \\ &29 \quad \quad K \leftarrow K + 1; \text{Insertion}(P_k, Newindividus); \\ &30 \quad \end{b}\end{aligned}$$

Algorithme 4 : Algorithme Génétique

2.6 Conclusion

Comme on a pu le constater, le comportement collectif des éléphants et des abeilles ainsi que la complexité de leur psychologie ont donné lieu à de puissants algorithmes de résolution orientés espaces des solutions. Ces derniers ont prouvé leur performance et leur efficacité face à de nombreux problèmes. Ainsi, vu notre problématique, qui rappelons-le, consiste à détecter des cibles dans un espace complexe inconnu à l'aide de robots mobiles, nous avons choisi ces algorithmes intelligents pour la résolution de notre problème compte tenu de leur robustesse. Le chapitre qui suit portera sur la modélisation de

notre solution sur laquelle se baseront nos différentes approches mono-BSO, multi-BSO, EHO ainsi qu'ESWSA.

Chapitre 3 : Modélisation du problème de recherche de cibles

3.1 Introduction

L'étape de modélisation d'un algorithme basé intelligence en essaim est centrale, elle requiert une bonne analyse faite au préalable, ce qui est particulièrement nécessaire lorsqu'il s'agit des problèmes complexes.

Ce chapitre est dédié à la description de notre modélisation et conception; celles-ci regroupent la représentation de l'espace de recherche, de la solution, de la fonction objectif ainsi que tous les autres composants de l'environnement de recherche. Par la suite, nous passerons en revue des travaux connexes relatifs aux stratégies d'évitement d'obstacles, avant d'enchaîner avec la stratégie choisie avec justification de ce choix. Enfin nous nous pencherons sur le principe de fonctionnement de l'algorithme génétique, qui sera utilisé comme outil de réglage de paramètres des méta-heuristiques de recherche de cibles.

3.2 Modélisation de l'environnement de recherche

Nous avons modélisé notre environnement de recherche par une représentation 2D dite "*grid based*", sous forme d'une matrice comme le montre la figure 3.1. Notre environnement est caractérisé par une surface préalablement fixée, afin que les bordures délimitent la zone de recherche. La forme de l'environnement est carrée telle que :

$$\text{Surface} = \text{Taille}_{\text{Coté}} * \text{Taille}_{\text{Coté}}. \quad (3.1)$$

Taille_{coté} : C'est la longueur d'un segment du carré.

Dans cette grille, chaque case contient une valeur ainsi qu'une position / coordonnée (x,y) , composée d'une abscisse (x) et d'une ordonnée (y). Plusieurs valeurs sont admissibles dans les cases, celles-ci permettent de distinguer les objets comme suit :

- **Cible** : Dans ce cas la valeur assignée à cette position est égale à **1**.
- **Portée d'une cible** : La valeur est bornée par **0** et **1**, ces derniers exclus.
(valeur(x,y) $\in]0,1[$)
- **Obstacle** : Sa valeur est égale à **-1**.
- **Zone neutre** : Prend la valeur **0**.

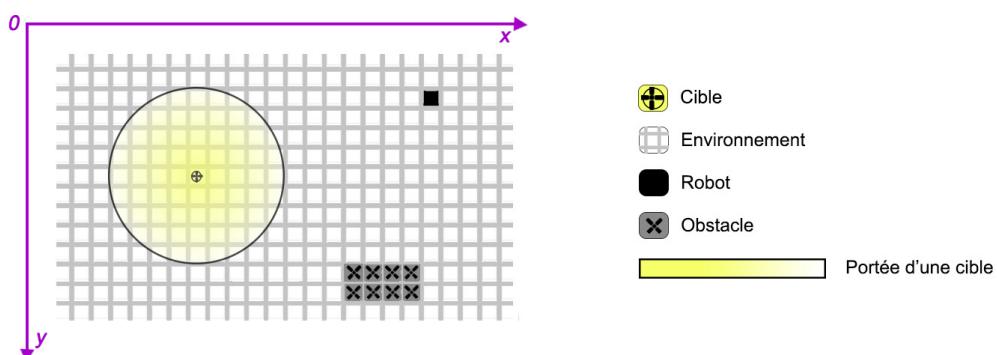


Figure 3.1: Représentation de l'environnement.

3.2.1 Représentation de la solution

Le but étant de trouver les cibles dans un environnement inconnu; l'espace des solutions à explorer n'est autre que notre environnement 2D composé de positions que nous appellerons aussi "solutions".

Une solution optimale est une position géographique de coordonnée (x, y) , elle est décrite par la valeur de sa case. La figure 3.2 représente une solution dans l'environnement de recherche.

Dans le cas de multiples cibles on devra trouver autant de solutions optimales que de cibles, telles que chaque solution correspond à une case de valeur 1.

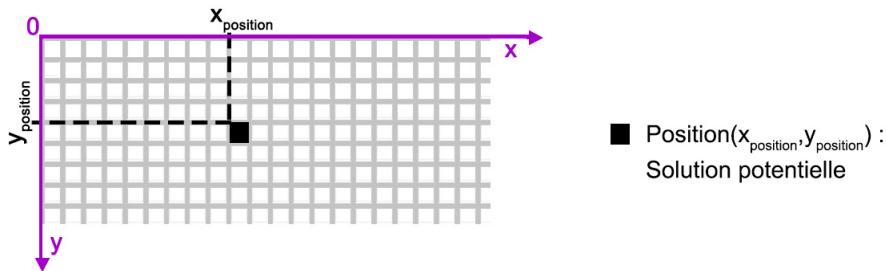


Figure 3.2: Représentation d'une solution dans notre modélisation.

Une solution doit satisfaire deux contraintes. D'abord, celle de la bornitude par les dimensions de l'environnement de recherche, ce dernier étant carré de taille $Taille_{Côté}$, une solution est telle que:

$$\begin{cases} 0 \leq Position.x < Taille_{Côté} \\ 0 \leq Position.y < Taille_{Côté} \end{cases} \quad (3.2)$$

Puis en tenant compte qu'une position contenant un obstacle n'est pas une solution admissible, soit :

$$environnement.get(Position.x, Position.y) \neq -1 \quad (3.3)$$

3.2.2 Fonction objectif

Les solutions ont besoin d'être évaluées pour mesurer leurs distances par rapport à une cible. Ainsi, on pourra choisir la plus proche à chaque itération.

Notre fonction objectif varie selon l'intervalle borné suivant : [0,1]. Elle est égale à la valeur captée par les capteurs du robot, telle que plus la valeur se rapproche du 1, plus le robot est proche de la cible, car son émission est plus importante. Comme nous le verrons plus bas (Section 3.3.3.2), l'évaluation se fait sur les cases visibles par le robot, cela à travers son champ de vision (comportant un certain nombre de cases).

3.2.3 Les cibles

Une cible occupe une case unique ayant une valeur de 1, elle possède aussi une portée circulaire fixe, relative à l'émission de signaux perceptibles par les robots.

Comme on peut le voir dans la figure 3.3, la portée décroît à partir de la position de la cible valant "1", jusqu'à la valeur "0" lorsque la distance par rapport à la cible excède la taille de la portée. Autrement dit, les valeurs représentatives des émissions de la cible sont inversement proportionnelles à la distance des positions incluses dans la portée.

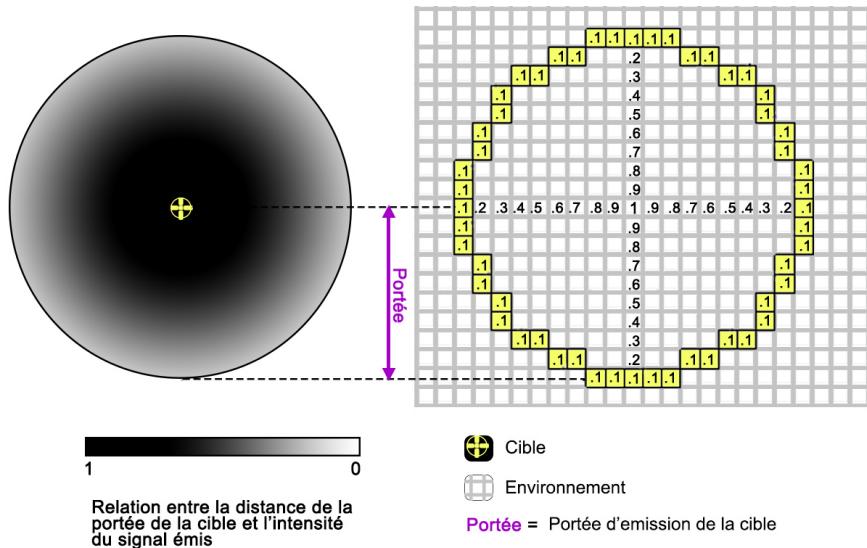


Figure 3.3: Représentation de la portée d'une cible dans notre modélisation.

Remarque: Une fois qu'une cible est atteinte par un robot, elle est désactivée, elle arrête alors d'émettre des signaux. Nous avons modélisé cette action par la mise à zéro des cases de la matrice de l'environnement constituant la portée de cette cible.

3.2.4 Les obstacles

Un obstacle est considéré comme un objet fixe (statique) occupant une certaine surface de l'environnement et empêchant les robots de s'y positionner ou de la traverser.

Les environnements peuvent être catégorisés en trois (3) types selon la densité des obstacles, soient:

- **Environnements simples (sans obstacles)** : Ceux-là ne contiennent que la ou les cible(s).
- **Environnements avec obstacles** : Comportant un nombre réduit d'obstacle de forme rectangulaire ainsi que la ou les cible(s).

La taille d'un obstacle est comprise entre 2% et 4% de la taille de l'environnement. Quant au nombre d'obstacles, il appartient à l'intervalle [15 , 25] obstacles.

- **Environnements complexes** : Ils possèdent plus d'obstacles qui sont plus grands en termes de superficie que ceux de la catégorie précédente, en plus de la ou les cible(s).

Dans ces environnements, la taille d'un obstacle est comprise entre 4% et 6% de la taille de l'environnement avec une densité de [25 , 35] obstacles.

Face aux différents types d'environnement existants, le choix d'une bonne stratégie d'évitement d'obstacles adaptée à notre modélisation devient crucial.

3.3 Stratégies d'évitement d'obstacles

Une stratégie d'évitement d'obstacles est nécessaire pour la navigation des robots dans des environnements à obstacles et complexes. Le but des méthodes d'évitement d'obstacles est de trouver une trajectoire sûre (sans obstacles) entre deux positions, c'est-à-dire entre la position initiale d'un robot et sa position destination ou but.

3.3.1 Description des approches existantes

Les techniques les plus utilisées pour l'évitement d'obstacles sont décrites ci-dessous:

3.3.1.1 Les algorithmes Bug [Sara, Bug]

Les algorithmes Bug1 et Bug2 sont des méthodes anciennes et simples. Ils réduisent le robot en un point dans un plan 2D détectant les obstacles via capteurs tactiles. Ils sont basés sur deux comportements : "déplacement vers le but" et "suivi d'une limite".

Bug1 : Tant que le robot n'a pas rencontré d'obstacle, il suit le comportement "déplacement vers le but", il se dirige tout droit vers la position but (T), lorsqu'il rencontre un obstacle en un point (H_i) il bascule vers le comportement "suivi d'une limite" en faisant un tour complet sur l'obstacle. À partir de là il détermine le point du périmètre de l'obstacle (L_i) le plus proche de la position but et s'y positionne. Ce processus se réitère jusqu'à atteindre la destination. (voir la figure 3.4).

Bug2 : Il exploite la première solution prometteuse qu'il trouve. Durant le comportement "déplacement vers le but", le robot se déplace selon des lignes droites reliant la position initiale (S) du robot et la position but (T). Quand un obstacle est rencontré, le robot adopte le comportement "suivi d'une limite", consistant à faire le tour de l'obstacle, jusqu'à atteindre un nouveau point (H_i) qui appartient à la droite (S, T) (voir la figure 3.4). Le processus est répété tant que la position but n'est pas atteinte par le robot.

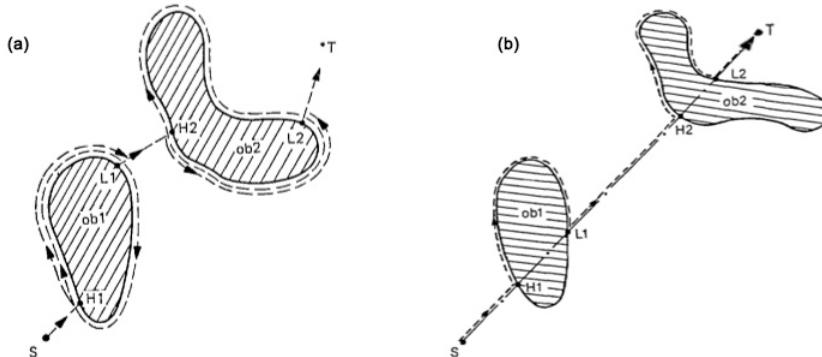


Figure 3.4: Calcul du chemin du robot par les algorithmes Bug : (a) Bug1, (b) Bug2 [Bug].

3.3.1.2 La méthode basée sur les champs de potentiel PF (Potential Field) [Sara]

Cette méthode a été proposée pour la première fois par Oussama Khatib [Khatib]. Elle considère le robot comme une particule plongée dans un champ de potentiel, celui-ci régie par deux forces, soient :

Force attractive $U_{attract}$: générée par le but.

Force répulsive $U_{répuls}$: générée par les obstacles.

Le robot calcule d'une manière itérative son prochain mouvement, suivant une direction résultante des sommes de différents champs potentiels (voir la figure 3.5). La direction est donnée par la fonction :

$$F = -\nabla (U_{attract} + U_{répuls}) \quad (3.4)$$

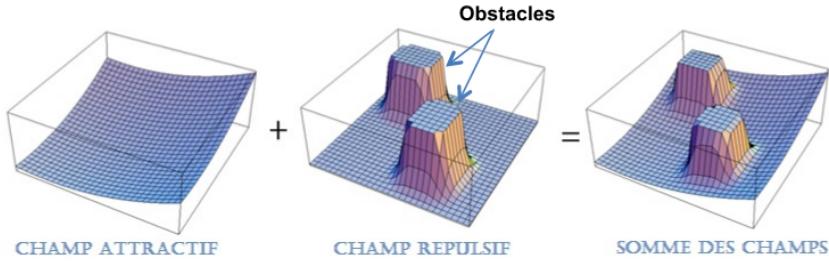


Figure 3.5: Champs de potentiel pour un environnement contenant deux obstacles et une position but.[ImagePF]

3.3.1.3 La méthode basée sur la fenêtre dynamique DW (Dynamic Window) [Sara]

La fenêtre dynamique est une méthode proposée par *Burgardand* et *Thruncite* [window]. Elle vise à choisir un couple (v, ω) représentant respectivement la vitesse linéaire et angulaire du robot, permettant d'éviter les obstacles perçus localement. Sur la base des différents couples possibles, DW opte pour le couple de vitesses le plus pertinent. Comme le montre la figure 3.6, la zone de recherche est limitée par une fenêtre dynamique basée sur les limitations dynamiques du robot et les vitesses admissibles et atteignables durant un laps de temps donné.

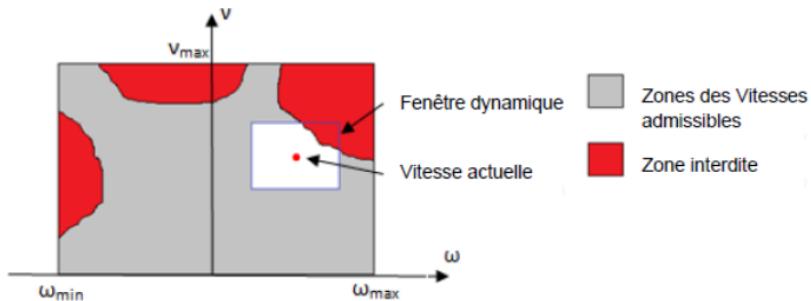


Figure 3.6: L'approche de la fenêtre dynamique montrant les régions admissibles et interdites par rapport à la fenêtre dynamique des vitesses atteignables par le robot [Sara].

3.3.1.4 La logique floue

La logique floue est une branche de l'intelligence artificielle et des mathématiques qui a été appliquée à des problèmes de commandes (Bühler 1997) [FuzzyLogic]. Cette technique se base sur la déduction de deux commandes : la vitesse linéaire et angulaire selon les variables floues des données en entrées, qui sont : la vitesse (v) du robot et l'angle (α) entre le robot et l'obstacle, en plus d'une base de règles afin d'éviter les obstacles [Cai2013].

3.3.1.5 L'échantillonnage de l'espace d'entrée ISS (Input Space Sample)

À partir d'un état initial du robot, le modèle prédictif du système est utilisé pour la génération d'un ensemble de trajectoires. Celles-ci peuvent être triées selon une fonction de coût. L'exemple d'un ensemble de trajectoires générées en utilisant la technique par échantillonnage de l'espace d'entrées est illustré dans la figure 3.7.

Selon les obstacles visibles par le robot, certaines trajectoires seront bannies de l'ensemble des trajectoires admissibles [Sara], comme il est le cas de $S1, S2, S7, S8$ et $S9$.

Les travaux de *Kelly* et *Stentz* [echantillonnage] font partie des premiers travaux basés sur cette technique.

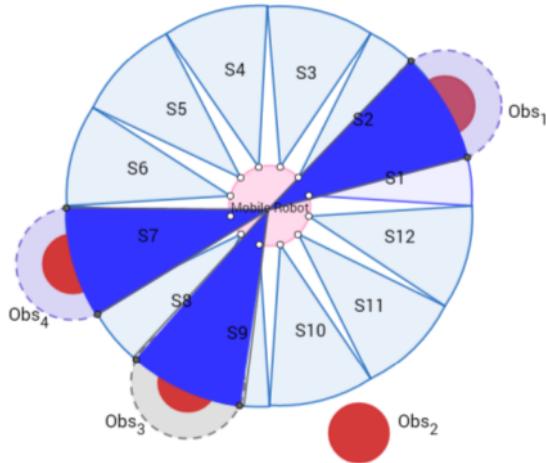


Figure 3.7: Ensemble de trajectoires générées par échantillonnage de l'espace d'entrées [imageEch].

3.3.2 Discussion et choix

Les algorithmes bug1 et bug2 sont faciles à mettre en œuvre et peu gourmands en termes de mémoires. Cependant, ils nécessitent des déplacements supplémentaires et souvent la trajectoire trouvée est loin d'être optimale (trajectoires comportant trop de détours supplémentaires augmentant ainsi la distance pour atteindre la position but).

Les méthodes de champ de potentiel et de logique floue, sont facilement adaptables aux formules des méta-heuristiques disposant d'une vitesse telles qu'ESWSA. En revanche, elles sont difficiles à mettre en pratique pour d'autres méthodes dont BSO et EHO, car le calcul des positions est indépendant de la direction de l'ancienne position vers la nouvelle.

De plus, elles ne permettent pas de détecter un passage entre des obstacles assez proches. Encore pire, le robot risque de ne jamais atteindre la position but, si ce dernier est à proximité directe d'un obstacle.

D'autre part, la méthode de la fenêtre dynamique (DW) n'est pas sujette aux limites précédemment citées. Mais elle peut être vu comme une recherche exhaustive d'une trajectoire optimale selon la dimension de la fenêtre, car pour une fenêtre de taille (x, y) nous aurons $x * y$ paires (v, w) à tester.

Enfin, la stratégie d'échantillonnage, n'est pas concernée par les problèmes cités ci-dessus dans le cadre de notre modélisation. Elle est simple et efficace, mais son plus grand avantage est qu'elle peut être directement adaptée ou intégrée dans d'autres concepts ou algorithmes.

Suite à l'analyse faite ci-dessous des avantages et inconvénients de chaque approche par rapport à notre modélisation, notre choix s'est porté sur la stratégie d'**échantillonnage** qu'on a jugé comme étant la plus adéquate.

3.3.3 Paramètres de la méthode d'échantillonnage

L'échantillonnage de l'espace local des robots est régi par plusieurs paramètres devant être fixés au préalable, soient :

3.3.3.1 Champ de vision

L'espace local d'un robot est divisé en quatre zones de 90° , comme le montre la figure 3.8 : "NE: Nord-Est, NW: Nord-Ouest, SW: Sud-Ouest, SE: Sud-Est".

Le champ de vision d'un robot est d'un angle de 90° , choisi selon la zone contenant la droite reliant la position du robot à la position but.

Afin de déterminer la zone concernée, nous devons effectuer deux opérations de soustraction sur les coordonnées du robot (x_R, y_R) et celles de la position destination (x_{but}, y_{but}). Les zones relatives aux résultats de ces opérations sont représentées dans le schéma ci-dessous.

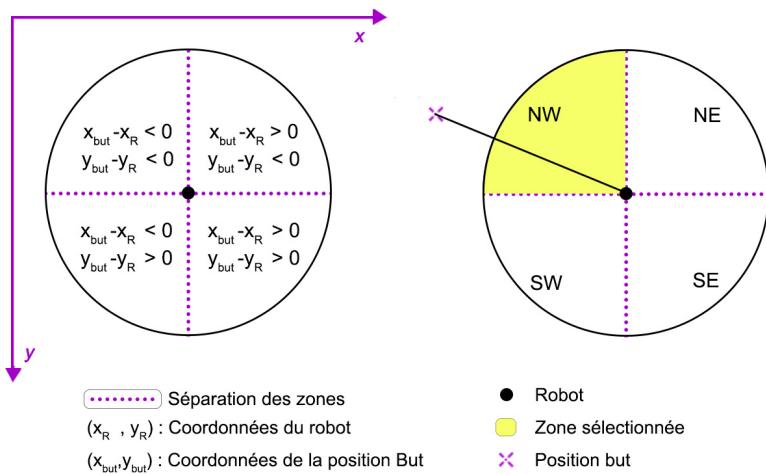


Figure 3.8: Détermination de l'angle de vue d'un robot dans notre modélisation.

3.3.3.2 Portée locale d'un robot

Chaque robot possède une portée locale, celle-ci forme une surface circulaire dont le robot est le centre. Le rayon de ce cercle est limité par la portée des capteurs présents sur le robot.

Dans notre modélisation, cette portée locale est fixée à une distance de dix positions du robot. La zone maximale visible par chacun de nos robots est représentée dans la figure 3.9 qui suit :

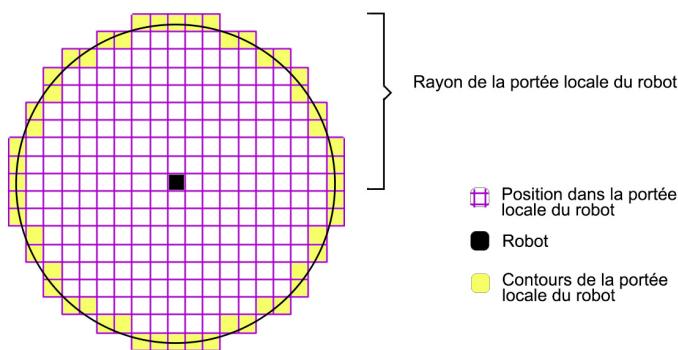


Figure 3.9: Représentation de la portée locale d'un robot dans notre modélisation.

Il est à noter que les robots ont cette portée de vue durant tout le parcours fait, c'est-à-dire que tout le long de leurs déplacements, une vérification de cette surface accessible est effectuée.

Nos robots prennent place dans les meilleures positions visibles dans leurs portées, cela à travers le processus suivant :

Input : P : position du robot,
 $portee$: Portée des capteurs = 10,
 env : Matrice de dimension $M \times M$

Réultat : meilleurePosition : Meilleure position maximisant la fonction objectif

```

1 meilleureLocale ← env.valeur( $P_x, P_y$ );
2 meilleurePosition ← Position( $P_x, P_y$ );
3 for  $i \leftarrow -portee$  to  $+portee$  do
4   for  $j \leftarrow -portee$  to  $+portee$  do
5     if env.valide( $P_x + i, P_y + j$ ) And  $i^2 + j^2 \leq portee^2$  then
6       b ← env.valeur( $P_x + i, P_y + j$ );
7       if meilleureLocale < b then
8         meilleureLocale ← b;
9         meilleurePosition ← Position( $P_x + i, P_y + j$ );
10      end
11    end
12  end
13 end
14 return meilleurePosition;
```

Algorithme 5 : Meilleure position dans la portée du robot

3.3.3.3 Trajectoire

Une trajectoire est modélisée par une droite reliant deux positions successives, cette droite est un ensemble de cases extraites selon l'équation de la droite, *Bresenham [line]* l'a exploité pour en déduire l'algorithme 6 suivant :

Input : x_0, y_0, x_1, y_1

```

1 dx = Abs ( $x_1 - x_0$ ); dy = -Abs ( $y_1 - y_0$ );
2 if  $x_0 < x_1$  then sx = 1;
3 else sx = -1;
4 if  $y_0 < y_1$  then sy = 1;
5 else sy = -1;
6 err = dx +dy;
7 while True do
8   setPoint ( $x_0, y_0$ );
9   e2 = 2*err;
10  if e2 >= dy then if  $x_0 == x_1$ 
11    then break;
12  err += dy;  $x_0 += sx$ ;
13  if e2 >= dx then if  $y_0 == y_1$ 
14    then break;
15  err += dx;  $y_0 += sy$ ;
16 end
```

Algorithme 6 : line-drawing algorithm [line]

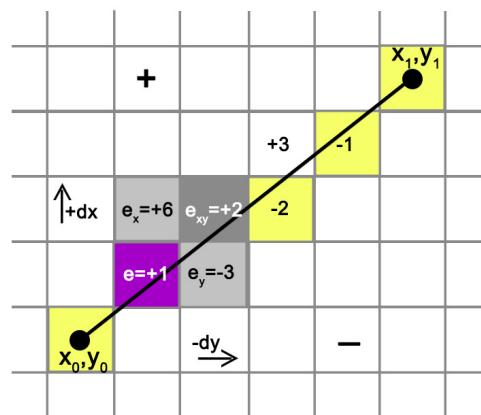


Figure 3.10: Sélection des points (cases) d'une droite avec l'algorithme de BRE-SENHAM.

Grâce à cet algorithme de Bresenham, nous pouvons au lieu de chercher à dessiner la droite à partir de deux positions, comme le montre la figure 3.10, l'exploiter pour vérifier si une trajectoire comporte des obstacles ou non.

3.3.3.4 Critères de choix de la trajectoire

Une fois que nous sommes en mesure de déterminer si une trajectoire est admissible (sans obstacles) ou interdite (contient des obstacles), vient l'étape de sélection de la trajectoire admissible la plus adéquate.

Cette seconde étape revient à choisir la meilleure trajectoire parmi celles admissibles, en minimisant la distance entre la position choisie (accessible) et la position destination de notre robot. Les deux étapes sont illustrées dans la figure 3.11 ci-dessous :

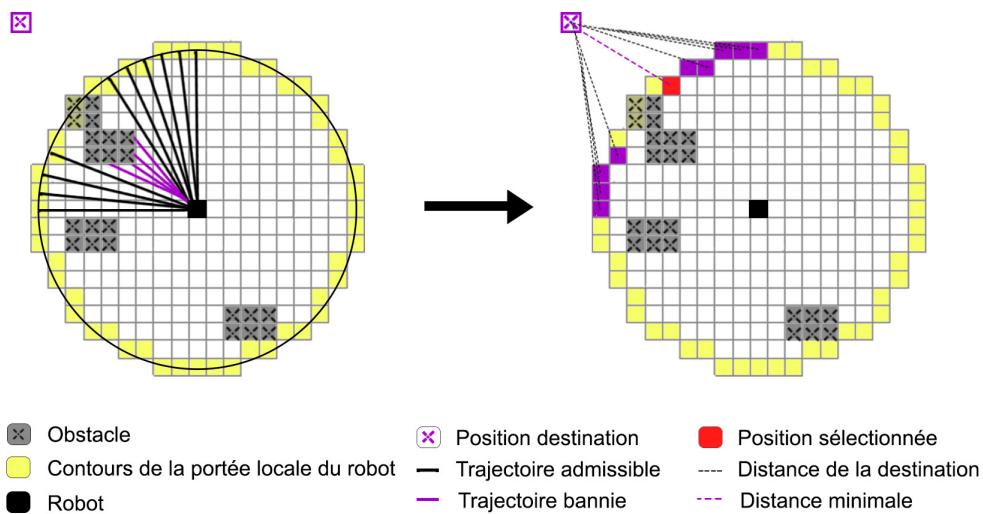


Figure 3.11: Choix de la trajectoire à suivre par le robot dans un environnement à obstacle.

Remarque: Si la position de destination est à une distance inférieure à la portée du robot (inférieure à dix cases.), le même procédé est effectué avec une portée égale à la distance entre le robot et la position destination.

Autres cas de figure Plusieurs cas de figure existent selon la position des obstacles par rapport à celle du robot, pour cela nous avons étudié toutes les possibilités pour garantir le bon fonctionnement de notre approche dans tous les environnements.

La figure 3.12 représente deux cas de figure se produisant souvent dans des environnements complexes. Lorsque aucune trajectoire admissible n'est trouvée dans le champ de vision initial du robot (90°), on l'élargit aux zones adjacentes de 90° allant ainsi à un champ de vision qui équivaut à 270° . Ceci simulera une rotation du robot sur lui-même (voir schéma de droite).

Si le robot ne trouve toujours pas de trajectoire admissible, une nouvelle position destination est demandée.

Dans le cas où la position destination est alignée sur un des deux axes avec la position actuelle du robot (c'est-à-dire qu'ils possèdent la même abscisse ou même ordonnée), notre robot aura une vue sur les 180° dont la droite (position du robot - position destination) est commune (schéma de gauche).

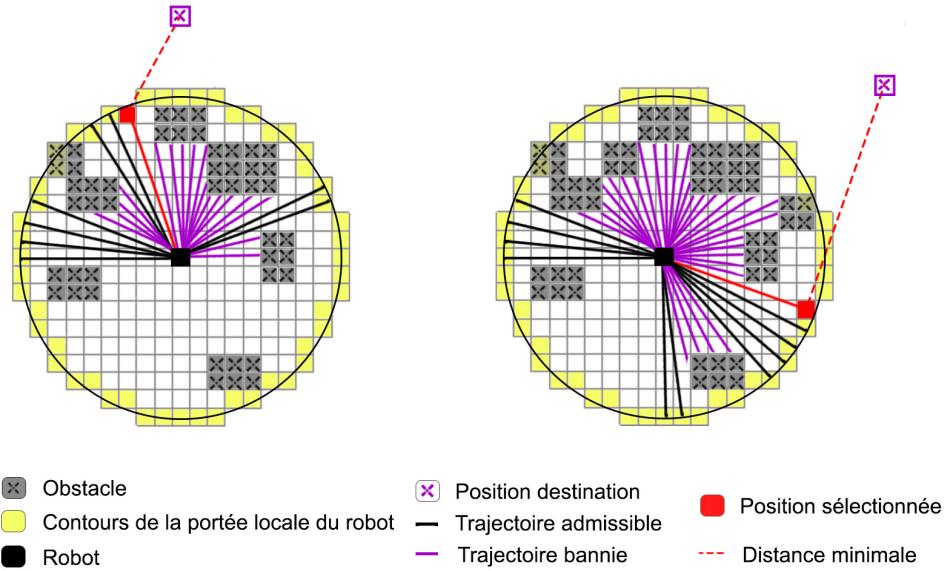


Figure 3.12: Choix de la trajectoire à suivre par le robot dans des situations complexes.

3.4 Auto-Paramétrage avec GA

3.4.1 Modélisation

3.4.1.1 Solution

Une solution pour notre algorithme génétique est un paramétrage pour une de nos approches de résolution. Elle est représentée par un vecteur de paramètres.

Vu que nos méta-heuristiques n'ont ni le même nombre de paramètres, ni les mêmes plages de valeurs de ces paramètres, la taille de la solution ainsi que les contraintes singulières relatives à chaque approche seront fixées, celles-ci sont représentées dans le tableau 3.1 ci-dessous :

Paramètres	Paramètre 1	Paramètre 2	Paramètre 3	Paramètre 4
BSO	Flip	nbrBees	MaxChance	-
Min	1	1	1	-
Max	100	25	4	-
Multi-BSO	Flip	nbrBees	MaxChance	nbSwarm
Min	1	1	1	1
Max	100	5	4	5
EHO	nbrClan	nbrElephant	Alpha	Beta
Min	1	1	0.1	0.1
Max	5	5	0.9	0.9
ESWSA	nbElephant	P	inertia Weight	-
Min	1	0.1	0.1	-
Max	25	0.9	0.9	-

Table 3.1: Contraintes sur les paramètres des méta-heuristiques.

3.4.1.2 Espace des solutions

L'espace des solutions diffère selon l'approche à paramétriser. Pour une approche donnée, Il s'agit de l'ensemble des combinaisons possibles des différentes valeurs de chaque paramètre de cette approche.

Comme présenté dans le tableau 3.1, les domaines de chaque paramètre sont clairement définis, ainsi l'espace des solutions de notre algorithme génétique comporte:

- Pour BSO : $100 \times 25 \times 4 = 10\,000$ solutions.
- Pour Multi-BSO : $100 \times 5 \times 4 \times 5 = 10\,000$ solutions.
- Pour EHO : $5 \times 5 \times 9 \times 9 = 2\,025$ solutions.
- Pour EWSA : $25 \times 9 \times 9 = 2\,025$ solutions.

3.4.1.3 Fonction objectif

Rappelons que l'objectif de notre algorithme génétique est de trouver la meilleure combinaison de paramètres pour nos approches de recherche de cibles. C'est pourquoi notre évaluation se résume en une exécution d'une approche de recherche. Compte à notre fonction objectif, elle se décompose en deux sous fonctions objectif:

1. Meilleure en termes de nombre de cibles trouvées, qui est à maximiser.
2. Meilleure en termes de nombre d'itérations, celle-ci est à minimiser.

Ces deux fonctions suivent une hiérarchie conforme à l'ordre dans lequel nous les avons citées. La meilleure solution est celle qui permet de trouver le maximum de cibles d'abord, mais aussi qui minimise le nombre d'itérations effectué pour les atteindre.

3.4.1.4 Population

La taille de la population N initiale est un paramètre à régler par expérimentations, il dépend aussi de l'espace des solutions à explorer.

3.4.1.5 Croisement

Un seul point de croisement est choisi, il est défini de manière aléatoire selon la contrainte suivante:

$$0 < \text{pointCroisement} < \text{nbrParamtres} \quad (3.5)$$

Une fois le point de croisement déterminé, nous passons à la formation des solutions fils, grâce à l'opérateur de concaténation. La figure 3.13 suivante explicite le déroulement d'un croisement.



Figure 3.13: Croisement de deux solutions.

3.4.1.6 Mutation

Une mutation est le remplacement d'un ou plusieurs paramètres de notre solution par une valeur aléatoire, toujours en respectant les contraintes d'intervalle de chaque paramètre cité dans le tableau 3.1. On choisit de faire une seule mutation aléatoire explicitée dans la figure 3.14 suivante:



Figure 3.14: Mutation d'une solution.

3.4.2 Fonctionnement

La méta-heuristique GA comporte une population de N individus dont nous essayons d'améliorer les gènes pour obtenir le meilleur individu au fil des générations. L'organigramme de la figure 3.15 décrit ce processus en détail.

Tout d'abord, les bornes (inférieurs et supérieurs) de chaque paramètre, pour chaque approche doivent être initialisées, afin que les valeurs générées pour chaque individu soient cohérentes avec l'espace des solutions. Puis, les N individus de notre population seront générés et évalués un à un selon la fonction objectif décrite plus haut, avant leur insertion dans la population P. L'auto-paramétrage suit les étapes suivantes :

3.4.2.1 Sélection des deux meilleures solutions

Les deux meilleures solutions de la population P de la génération "*t^{ième}*" sont sélectionnées, soient "S1" et "S2".

Ces deux parents sont supprimés de la population après sélection.

3.4.2.2 Croisement des deux solutions (parents)

On effectue lors de cette phase un croisement sur "S1" et "S2" selon le point de croisement **pointCroisement**, produisant ainsi deux autres solutions enfants: "S1_{fils}" et "S2_{fils}".

3.4.2.3 Évaluation des solutions fils

On évalue ensuite les solutions enfants "S1_{fils}" et "S2_{fils}" selon la fonction objectif (nombre de cibles trouvées et nombre d'itérations).

3.4.2.4 Mutation des solutions fils

Les solutions fils subiront un nombre **nbrMutation** de mutations. Il en résultera deux autres solutions mutantes, qu'on notera : "S1'_{fils}" et "S2'_{fils}".

3.4.2.5 Évaluation des solutions mutantes

Comme pour les solutions fils, les solutions mutantes seront évaluées selon la fonction objectif.

3.4.2.6 Injection des solutions résultantes dans la population

Une fois les quartes solutions évaluées à savoir : "S1_{fils}", "S2_{fils}", "S1'_{fils}" et "S2'_{fils}", on les insère dans la population courante.

3.4.2.7 Passage à la génération suivante

Après avoir enrichie la population avec les nouveaux individus, on incrémente l'indice de génération " t ", afin de passer à la génération suivante.

3.4.2.8 Critère d'arrêt

Toutes ces étapes du processus de paramétrage par l'algorithme génétique, seront réitérées jusqu'à ce que le nombre de générations **MaxGen** soit atteint.

Pour notre solution finale, nous sélectionnons les cinq meilleures solutions de la population, pour en calculer la moyenne.

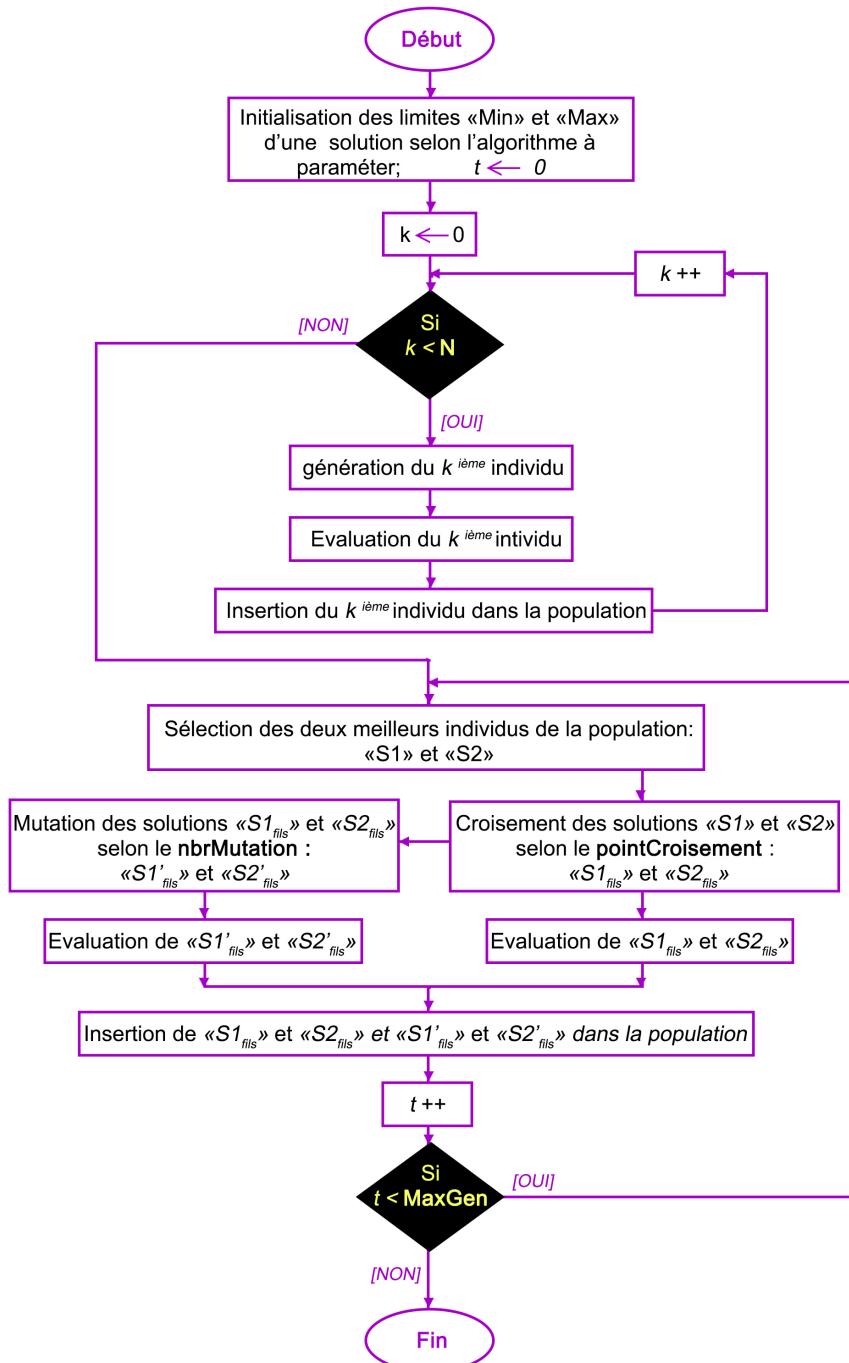


Figure 3.15: Organigramme du mode de fonctionnement de la méthode GA.

3.5 Conclusion

Ce chapitre a fait l'objet de la modélisation de tous les aspects de notre problème, c'est à partir de cette modélisation que nous avons choisie une stratégie d'évitement d'obstacles applicable à toutes nos méthodes de résolution basées essaims. Tous les détails relatifs à son fonctionnement y sont présentés. Enfin, nous avons explicité l'usage de l'algorithme Génétique pour le paramétrage automatique. Ces parties communes à l'ensemble de nos approches de recherche de cibles définies, nous nous attelons à présenter nos approches de recherche de cibles, commençant par BSO dont il est question dans le chapitre suivant.

Chapitre 4 : BSO pour le problème de recherche de cibles

4.1 Introduction

Dans ce chapitre, nous présentons notre approche de résolution basée sur l'algorithme BSO. Dans un premier temps, le robot simule le comportement d'une seule abeille alors que dans une seconde implémentation, le robot simule tout un essaim d'abeilles, l'approche dans ce cas est appelée Multi-BSO. Le principe de fonctionnement de chaque algorithme, à savoir BSO, Multi-BSO sera détaillé.

Un récapitulatif du travail effectué et de son importance pour la phase d'implémentation clôtura ce chapitre.

4.2 Mono-BSO vs Multi-BSO

Les robots constituent un seul groupe qui simule le comportement des abeilles. Il existe donc une communication de type stigmergique entre les robots, concrétisée à l'aide de la simulation de la danse des abeilles. Ce qui amène à déduire qu'il existe une intelligence collective entre les robots déployés et qu'ils interagissent de manière coopérative pour la recherche des cibles.

Nos deux approches BSO et Multi-BSO reposent certes toutes les deux sur les mêmes concepts de base, mais sont utilisées de manières bien différentes, telles que pour :

- **Mono-BSO**, chaque robot se comporte comme une seule abeille. Il occupe une case ayant une certaine position dans l'environnement et peut tester la qualité de sa position en tant que solution, grâce à la fonction objectif.
- **Multi-BSO**, comme son nom l'indique consiste en l'application de BSO par plusieurs essaims d'abeilles. Dans ce cas, chaque robot se comporte comme un essaim d'abeilles lui permettant de se déplacer à la recherche d'une solution optimale.

4.3 Adaptation de BSO pour le problème de la recherche de cibles

Au deuxième chapitre, nous avons exposé l'algorithme générique BSO pour la résolution d'un problème complexe quelconque. Dans cette section, nous nous intéressons à son application au problème de la recherche de cibles. Chaque composant spécifique à l'algorithme comme la solution, la fonction objectif, le paramètre Flip et la table Dance sera adapté à ce problème.

4.3.1 Solution

Une solution est la position de coordonnées (x, y) dans laquelle se trouve une abeille. Cette position correspond à une case ayant une certaine valeur.

4.3.2 Fonction objectif

Chaque abeille est responsable de l'évaluation de sa solution, en vérifiant la valeur contenue dans la case où elle se trouve. Si une cible est détectée la valeur trouvée est alors de 1.

4.3.3 Flip

Il s'agit du paramètre de diversification. Dans l'environnement que nous traitons pour ce problème, le **flip** permet de déterminer la zone de recherche globale à partir d'une position initiale I , cela à travers le calcul des positions à une distance R (rayon) de la position initiale.

$$R = \text{TailleCoté} / \text{flip} \quad (4.1)$$

Avec: TailleCoté : Taille du coté de l'environnement.

Les positions générées constituent le cercle de rayon R ayant pour centre la position initiale I . La figure 4.1 schématisé ce processus.

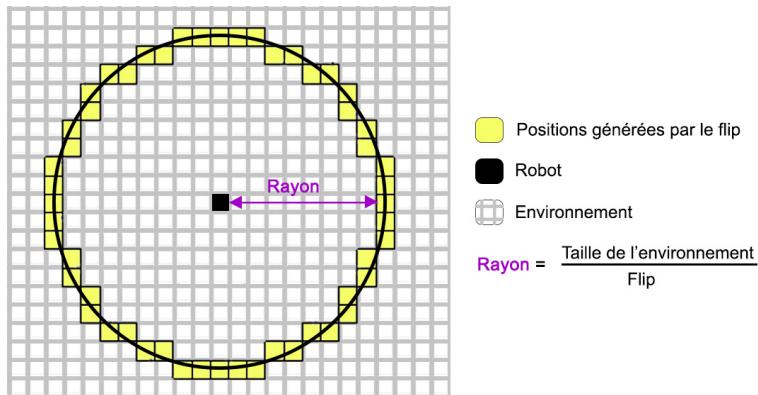


Figure 4.1: Représentation de la méthode de génération de solutions avec le flip.

4.3.4 MaxChance

Une solution possède un nombre maximum de chances pour être sélectionnée comme solution de référence. Ce paramètre est aussi exploité afin de détecter la stagnation.

4.3.5 Table Dance

À chaque itération les abeilles déposent leurs meilleures solutions dans une table appelée "*Table Dance*". Cette table sera exploitée pour choisir la solution de référence de la prochaine itération. Ce choix se fait selon l'un des deux critères suivants:

4.3.5.1 Critère de qualité (Best In Quality)

Parmi les solutions présentes dans la table **Dance**, la meilleure en termes de qualité est celle possédant la valeur la plus élevée après évaluation. Un exemple de sélection de la meilleure solution est illustré dans la figure 4.2 suivante:

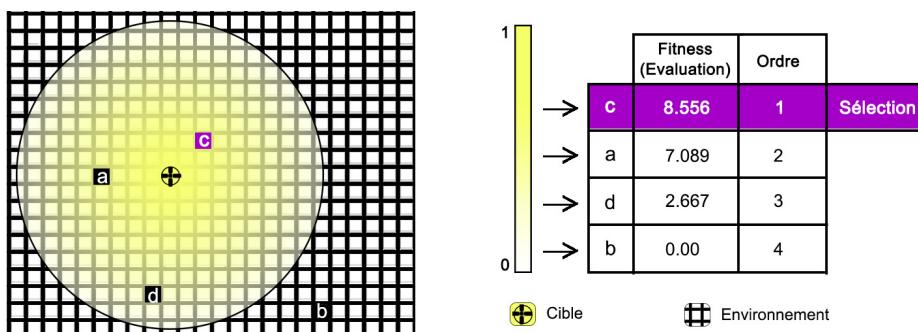


Figure 4.2: Méthode de sélection de la meilleure solution en termes de qualité.

4.3.5.2 Critère de Diversité (Best In Diversity)

En cas de stagnation, BSO a recours au choix de la meilleure solution en termes de diversité appartenant à la table *Dance* et ne figurant pas encore dans la liste *Tabou*. Ce mécanisme permet l'exploration des zones distantes tout en évitant de tomber dans un minimum local.

Pour cela nous devons calculer la distance entre deux solutions, la formule choisie est la distance euclidienne, donnée comme suit:

$$\text{Distance}(S_1, S_2) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \quad (4.2)$$

S_1 : Solution 1 avec coordonnées (x_1, y_1)

S_2 : Solution 2 avec coordonnées (x_2, y_2)

Le choix de la solution la plus diverse se fait en sélectionnant la solution dont la distance minimale par rapport aux autres solutions est maximale. Cette méthode de sélection est schématisée dans la figure 4.3 ci-contre:

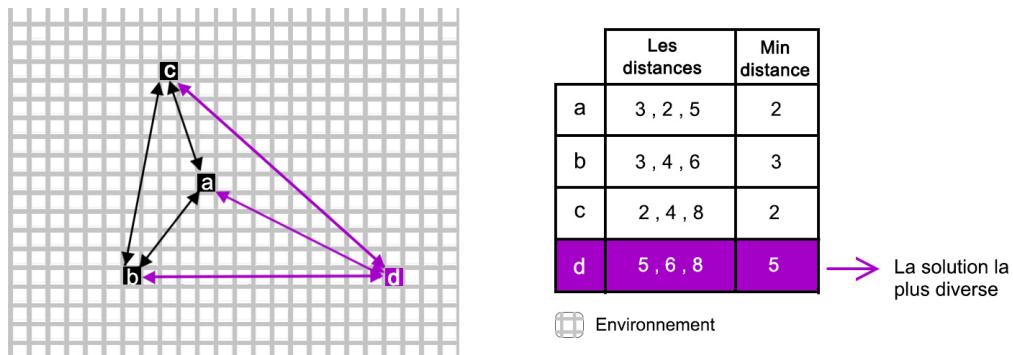


Figure 4.3: Méthode de sélection de la meilleure solution en termes de Diversité.

4.4 Fonctionnement du Mono-BSO

L'approche Mono-BSO est constituée d'un seul essaim d'abeilles où chaque robot se comporte comme une seule abeille. Les différentes étapes du Mono-BSO pour la recherche de cibles sont détaillées dans ce qui suit, suivi d'un organigramme 4.4 résumant ces derniers.

4.4.1 Initialisation de la solution de référence (Sref)

Une solution ($\text{position}(x,y)$) est générée aléatoirement et conformément aux contraintes énoncées dans notre modélisation (voir section 3.2.1).

4.4.2 Insertion de Sref dans la liste Tabou

La solution de référence (Sref) de l'itération courante " t " sera mise dans une liste *Tabou* qu'on définit comme suit:

Liste Tabou Lorsqu'une position a déjà servi pour une abeille, nous gardons trace son passage dans la liste "*Tabou*", cela permet de réduire la redondance (passage par les mêmes positions).

4.4.3 Génération des zones de recherche

À partir de la solution de référence (Sref) on génère d'autres solutions, ces dernières déterminent des zones équidistantes de Sref. La génération de ces zones de recherche se fait grâce à l'opérateur **Flip**.

4.4.4 Affectation des zones aux abeilles

Une fois les zones générées, on les affecte aux "**nbrBees**" abeilles de l'essaim de façon à ce que chaque abeille soit responsable de sa propre zone.

4.4.5 Recherche locale pour chaque abeille

Chaque abeille effectue une recherche locale qui consiste en l'évaluation d'une surface ronde de l'environnement dont le rayon est de vingt cases, cela constitue l'étape d'intensification.

Dans le cas où la recherche locale apporte une amélioration, l'abeille prend cette nouvelle solution et réitère ce processus jusqu'à l'absence d'amélioration.

4.4.6 Déplacement des abeilles

Lors de la recherche locale, les abeilles peuvent trouver de meilleures solutions que celles qui leur ont été affectées, c'est pourquoi elles se déplacent vers ces solutions plus prometteuses dans l'environnement, ce qui simule le déplacement du robot. Chaque déplacement d'une position à une autre se fait à travers notre stratégie d'évitement d'obstacles (stratégie d'échantillonnage).

4.4.7 Insertion des meilleures solutions dans la table *Dance*

Lorsqu'une abeille termine sa recherche locale elle communique sa meilleure solution au reste de l'essaim, cela en l'inscrivant dans la table *Dance* commune à l'essaim.

Lorsqu'une cible est atteinte on incrémente le compteur relatif au nombre de cibles trouvées, si le nombre objectif de cibles "**nbrCible**" est égalé, la recherche prend fin avec succès.

4.4.8 Choix de la nouvelle Sref

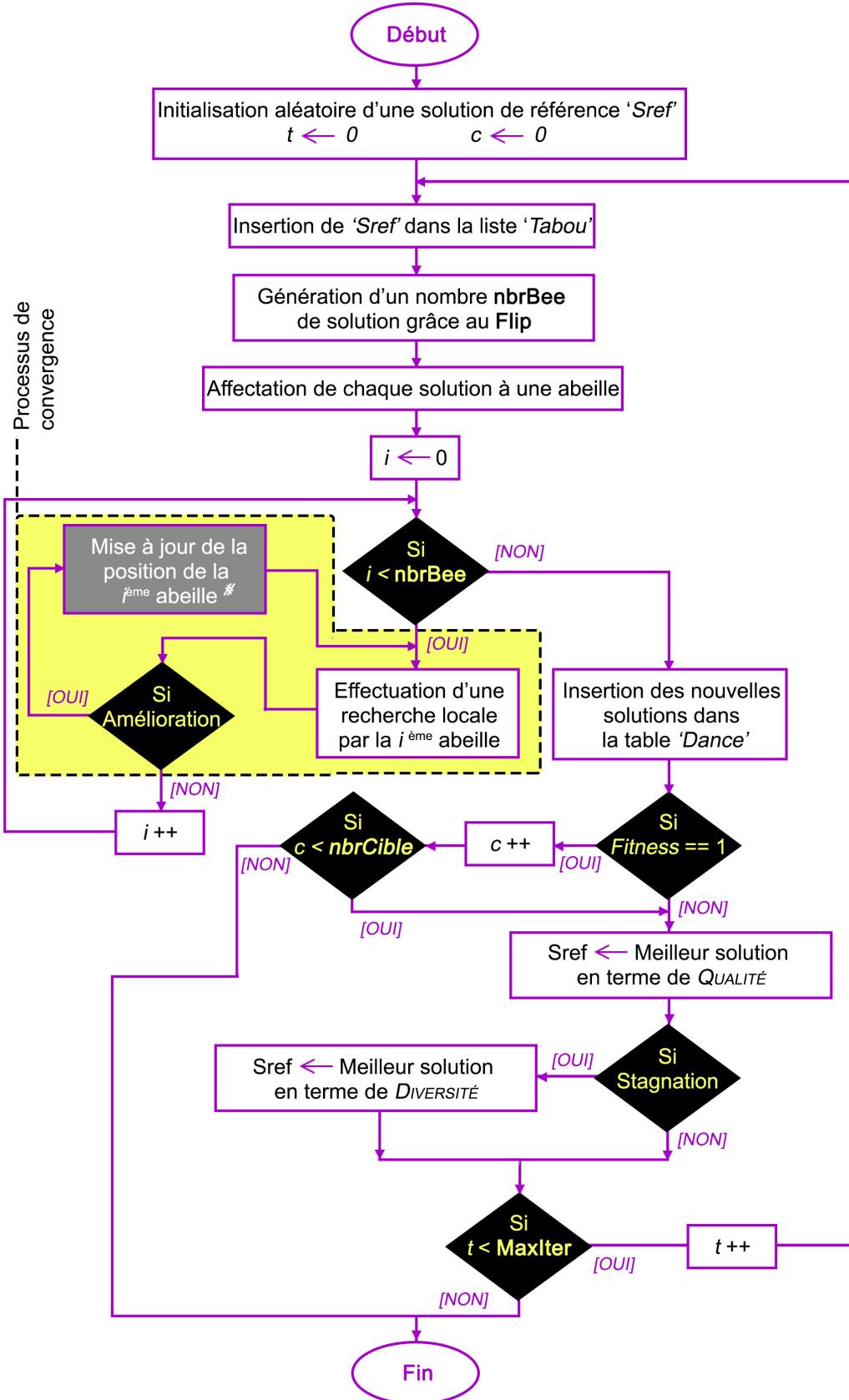
Si à l'itération courant "*t*" le nombre de cibles recherchées n'est pas atteint, une nouvelle solution de référence doit être choisie pour la prochaine itération.

Il existe alors deux critères possibles, le critère prioritaire est celui de qualité. Une solution prise selon le critère de qualité peut demeurer comme Sref au fil des itérations pour un nombre "**MaxChances**" de fois. Dans le cas où toutes les chances données à cette Sref sont épuisées (stagnation), le choix de la nouvelle Sref se fera selon le second critère, celui de diversité.

4.4.9 Critère d'arrêt de la recherche

BSO continuera de réitérer les étapes citées plus haut à partir de 4.4.2 jusqu'à atteindre le nombre maximum d'itérations "**MaxIter**" ou le nombre de cibles recherchées "**nbrCible**".

Remarque: Contrairement aux autres méta-heuristiques, BSO n'utilise pas seulement la fonction d'évaluation comme critère de sélection de solution, mais aussi la diversité. C'est un mécanisme pour remédier au problème de stagnation dans les minima locaux.



* Appel de la stratégie d'évitement d'obstacle avec l'ancienne et la nouvelle position calculée

Figure 4.4: Organigramme du mode de fonctionnement de l'approche mono-BSO.

4.5 Fonctionnement du Multi-BSO

L'approche Multi-BSO quant à elle, consiste à associer à chaque robot, un essaim d'abeilles. Ils sont donc indépendants les uns des autres, mais ils sont en mode de coopération pour la recherche des cibles. Ils suivent le modèle d'interaction du tableau noir des systèmes multi-agents. Le processus de recherche de cibles implémenté à l'aide de Multi-BSO est décrit dans ce qui suit :

4.5.1 Initialisation des solutions de référence (Srefs)

Un nombre "**nbrSwarm**" de solutions (x,y) est généré aléatoirement dans l'espace des solutions, de telle sorte à respecter les contraintes citées dans notre modélisation (voir section 3.2.1). Chaque position Sref est attribuée à un robot.

4.5.2 Insertion de Sref dans la Liste Tabou

Chaque essaim d'abeilles relatif à un robot dépose sa solution de référence (Sref) dans la liste *Tabou*. Cette Liste *Tabou* est commune à tous les robots, elle constitue leur moyen de communication (inter-essaim), permettant de réduire la redondance et repassage sur les mêmes solutions.

4.5.3 Génération des zones de recherche

À partir de la solution de référence (Sref) de chaque robot, les zones de recherche sont générées grâce à l'opération de "*Flip*". Chaque zone est définie par une position dans l'espace de recherche.

4.5.4 Affectation de chaque zone à une abeille

Pour chaque robot disposant de "**nbrBees**" abeilles, chaque zone de recherche calculée précédemment est affectée à une abeille de son essaim, celle-ci y prend position.

4.5.5 Recherche locale pour chaque abeille

Chaque abeille effectue une recherche locale à partir de la zone qu'elle s'est vue affectée. Cette recherche consiste en l'évaluation des solutions (positions) de son voisinage à travers la fonction objectif.

À noter que tant qu'une abeille améliore la qualité de sa solution la recherche locale est réitérée de manière récursive.

4.5.6 Insertion des solutions dans les tables *Dance*

Chaque essaim possède sa propre table *Dance*, celle-ci permet la communication intra-essaim de telle sorte que chaque abeille y insère sa meilleure solution locale (de sa zone), afin d'en informer le reste de l'essaim.

4.5.7 Test de la 1^{ère} condition d'arrêt

Si une des abeilles a atteint une cible (solution optimale), le nombre de cibles trouvées "*c*" est mis à jour. Dans le cas où le nombre de cibles recherchées "**nbrCible**" est égalé, alors la mission touche à sa fin.

4.5.8 Choix de la nouvelle solution de référence

À priori, la meilleure solution en matière de qualité est sélectionnée à partir de la table *Dance* comme nouvelle solution de référence (Sref). Sauf en cas de stagnation où une même solution est prise pour solution de référence au-delà de "MaxChances" fois, Sref est alors choisie selon le critère de diversité.

4.5.9 Déplacement des robots

Chaque robot se déplace de son ancienne position vers la nouvelle position calculée (Sref) en employant notre stratégie d'évitement d'obstacles.

4.5.10 Test de la 2^{ème} condition d'arrêt

Le nombre d'itérations " t " est incrémenté puis comparé au nombre maximum d'itérations "MaxItér" accordé à la recherche.

L'approche Multi-BSO peut alors être perçue comme l'application de Mono-BSO par chaque robot. L'organigramme de son fonctionnement est résumé dans la figure ci-contre.

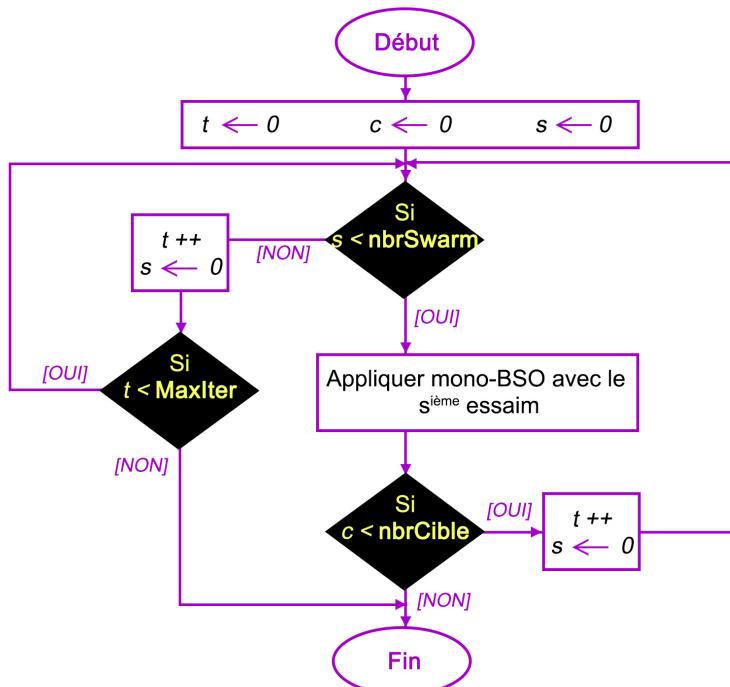


Figure 4.5: Organigramme du mode de fonctionnement de l'approche multi-BSO.

4.6 Conclusion

À travers ce chapitre, on a pu mettre en évidence deux approches inspirées du comportement des abeilles BSO et Multi-BSO. Ces algorithmes intelligents reposent sur les mêmes paramètres et fonctions de base, mais avec un mode d'emploi légèrement différent pour Multi-BSO qui nécessite la coordination des essaims (équipes) d'abeilles.

Tout en restant dans la même dynamique des techniques inspirées des comportements d'animaux, nous nous penchons dans le chapitre suivant sur deux approches simulant le comportement des éléphants, qui sont très différents des abeilles de par leurs tailles et mécanismes sociaux.

Chapitre 5 : Algorithmes des éléphants pour le problème de recherche de cibles

5.1 Introduction

La deuxième contribution majeure de ce projet est l'adaptation des deux algorithmes basés essaim d'éléphants vus précédemment, à savoir EHO et EWSA pour le problème de la recherche de cibles. Il sera question dans ce chapitre d'effectuer une adaptation de l'ensemble des concepts de ces deux méthodes de résolution à la modélisation exhibée au niveau du chapitre 3, ainsi que les différentes étapes de leur fonctionnement.

5.2 Adaptation de EHO pour le problème de la recherche de cibles

L'algorithme EHO fut présenté de manière générale au niveau du chapitre 2, nous allons maintenant nous intéresser à son adaptation au problème de recherche de cibles. Les aspects phares de cette approche seront redéfinies comme suit :

5.2.1 Solution

Une solution est la position de coordonnées (x, y) dans laquelle se trouve un éléphant.

5.2.2 Fonction objectif

La fonction objectif est comme définie pour l'approche BSO, à la différence près que pour EHO ce sont les éléphants qui sont responsables de l'évaluation des solutions.

5.2.3 Éléphant

Un éléphant est un robot, il occupe une case unique dans l'environnement à chaque instant " t ". La mise à jour de la position d'un éléphant par rapport à son clan suit l'équation 2.1 pour les deux dimensions (x,y) , comme suit :

$$\begin{aligned} x_{new,c,i} &= x_{c,i} + \alpha * (x_{Best,c} - x_{c,i}) * r \\ y_{new,c,i} &= y_{c,i} + \alpha * (y_{Best,c} - y_{c,i}) * r \end{aligned} \quad (5.1)$$

La figure 5.1 ci-dessous représente l'influence de la meilleure solution sur le déplacement des autres éléphants du clan selon différents degrés α .

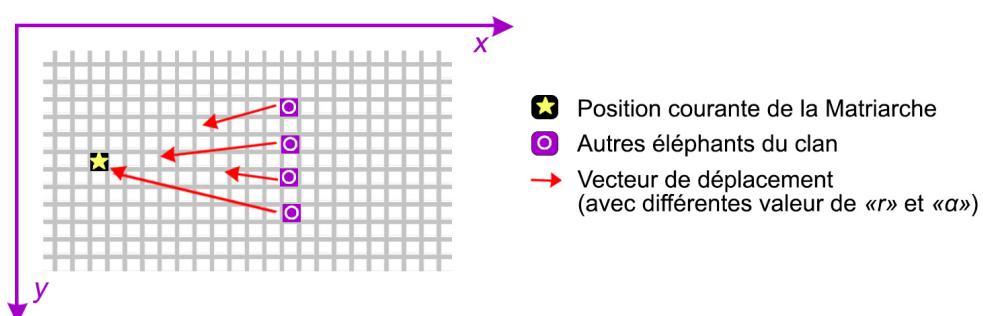


Figure 5.1: Représentation de la méthode de mise à jour des positions des éléphants.

5.2.4 Clan

Les éléphants d'un même clan sont généralement regroupés dans une même zone. Ainsi, les positions initiales des éléphants sont générées comme suit:

1. Pour chaque clan générer une position initiale aléatoirement.
2. À partir de chaque position générée, positionner les **nbrElephant - 1** autres éléphants sur une surface ronde de rayon inférieur ou égale à une certaine limite (5 cases dans notre cas).

La figure 5.2 ci-dessous, représente un exemple de positions générées pour les éléphants d'un même clan :

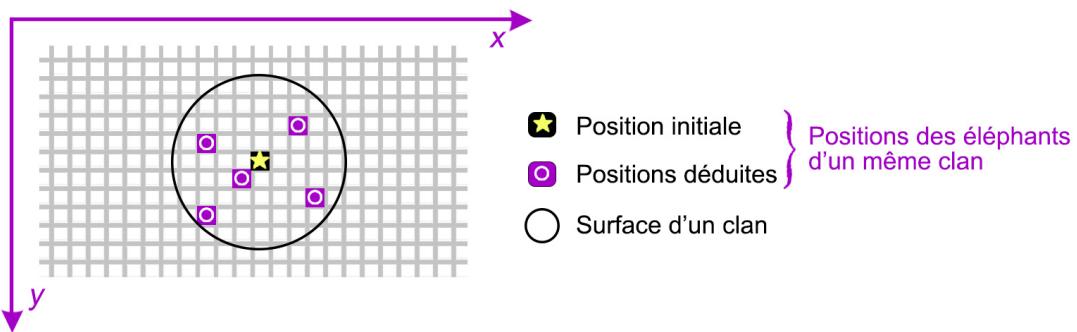


Figure 5.2: Représentation de la technique de génération des positions des éléphants d'un même clan.

5.2.5 Chef de clan (Matriarche)

L'éléphant chef de clan appelé "Matriarche" est celui qui possède la meilleure solution de son clan. La mise à jour de cette position dépend du centre de gravité du clan, comme présentée dans l'équation 2.2, elle est appliquée de la même manière à chacune des coordonnées x et y , comme suit :

$$\begin{aligned} x_{Best,c} &= \beta * x_{center,c} \\ y_{Best,c} &= \beta * y_{center,c} \end{aligned} \quad (5.2)$$

De même, le centre de gravité est calculé pour les deux dimensions, par l'équation:

$$\begin{aligned} x_{center,c} &= \frac{1}{N} * \sum_{i=1}^N x_{c,i} \\ y_{center,c} &= \frac{1}{N} * \sum_{i=1}^N y_{c,i} \end{aligned} \quad (5.3)$$

La figure 5.3 montre le processus de mise à jour de la position de la "Matriarche".

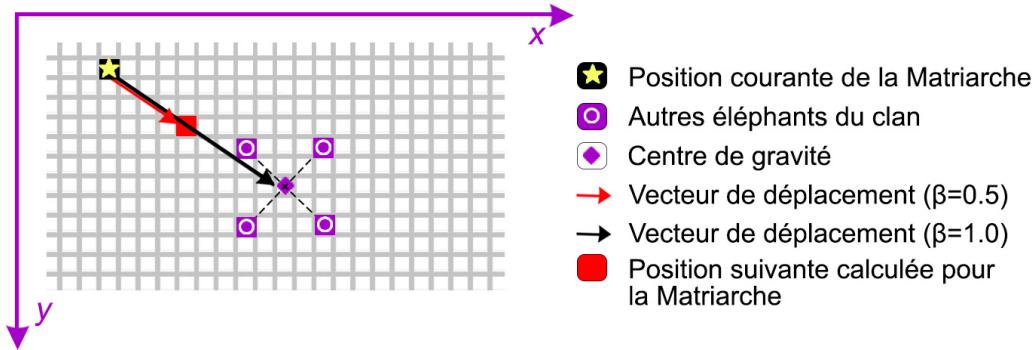


Figure 5.3: Représentation de la méthode de mise à jour de la position du meilleur éléphant/robot.

5.2.6 Éléphant mâle

Dans chaque clan, l'éléphant évalué comme détenant la pire solution est appelé "*pire éléphant*". Il possède une fonction de mise à jour particulière décrite par l'équation 2.4, qui pour notre environnement 2D de taille $taille_{Coté}$ est adaptée par l'équation :

$$\begin{aligned} x_{Worst,c} &= (taille_{Coté} - 1) * rand \\ y_{Worst,c} &= (taille_{Coté} - 1) * rand \end{aligned} \quad (5.4)$$

Celle-ci favorise la diversification afin d'explorer une plus grande surface de l'environnement. Un exemple d'application de cette équation est présenté dans la figure 5.4 qui suit :

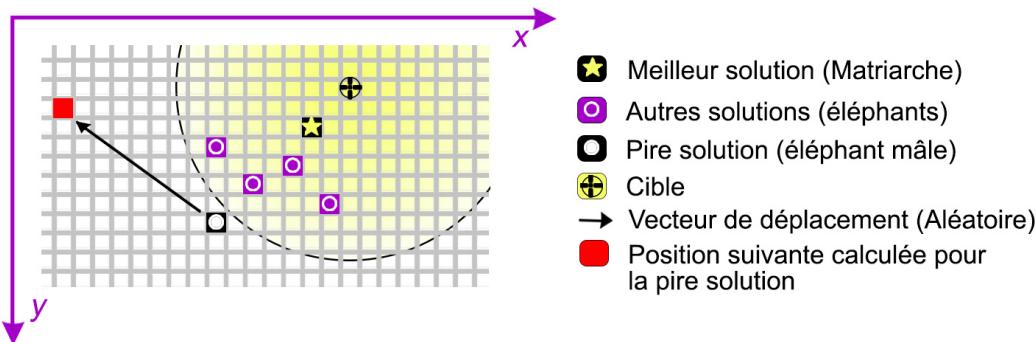


Figure 5.4: Représentation de la méthode de mise à jour de la position du pire éléphant/robot.

5.3 Fonctionnement d'EHO

La méthode EHO repose sur la présence de plusieurs clans de robots éléphants, où chaque clan possède un chef aussi appelé "*meilleur éléphant*" ou "*Matriarche*" et un pire éléphant devant se séparer du clan. Les différents clans coopèrent dans le but commun de recherche de cibles. Deux types de communication sont utilisés, d'abord la communication entre robots d'un même clan sur les meilleures solutions trouvées, en second lieu la communication entre clans relative au nombre de cibles, celle-ci faisant office de tableau noir. L'organigramme de la figure 5.5 décrit son fonctionnement.

5.3.1 Initialisation des positions des clans

Pour commencer, nous initialisons les positions des "**nbrClan**" clans de manière aléatoire dans notre espace des solutions, en tenant compte des contraintes concernant les bornes et les obstacles de ce dernier.

5.3.2 Initialisation des positions des éléphants

À partir des positions des clans, sont déduites les positions des "**nbrEle**" éléphants de chaque clan comme décrit dans la section 5.2.4.

5.3.3 Évaluation des solutions

Chaque clan " Cl " se voit évaluer les solutions que portent tous ses éléphants. Dès qu'une nouvelle cible est trouvée, le nombre de cibles détectées " c " est incrémenté. Ainsi, si le nombre total de cibles "**nbrCible**" de notre environnement est atteint, la mission de recherche est un succès et donc interrompue.

5.3.4 Tri des clans

Chaque clan parmi les "**nbrClan**", procède au tri de ses éléphants selon la fonction objectif dans l'ordre décroissant de ses valeurs.

5.3.5 Mise à jour des positions des éléphants

Pour chaque éléphant " ele " du clan " Cl " on détermine la nouvelle position. Pour cela, les deux paramètres nécessaires sont la position du chef de clan et le paramètre empirique α .

La nouvelle position du meilleur éléphant de chaque clan est calculée à partir du centre de gravité du clan CG_{Cl} et du paramètre empirique β .

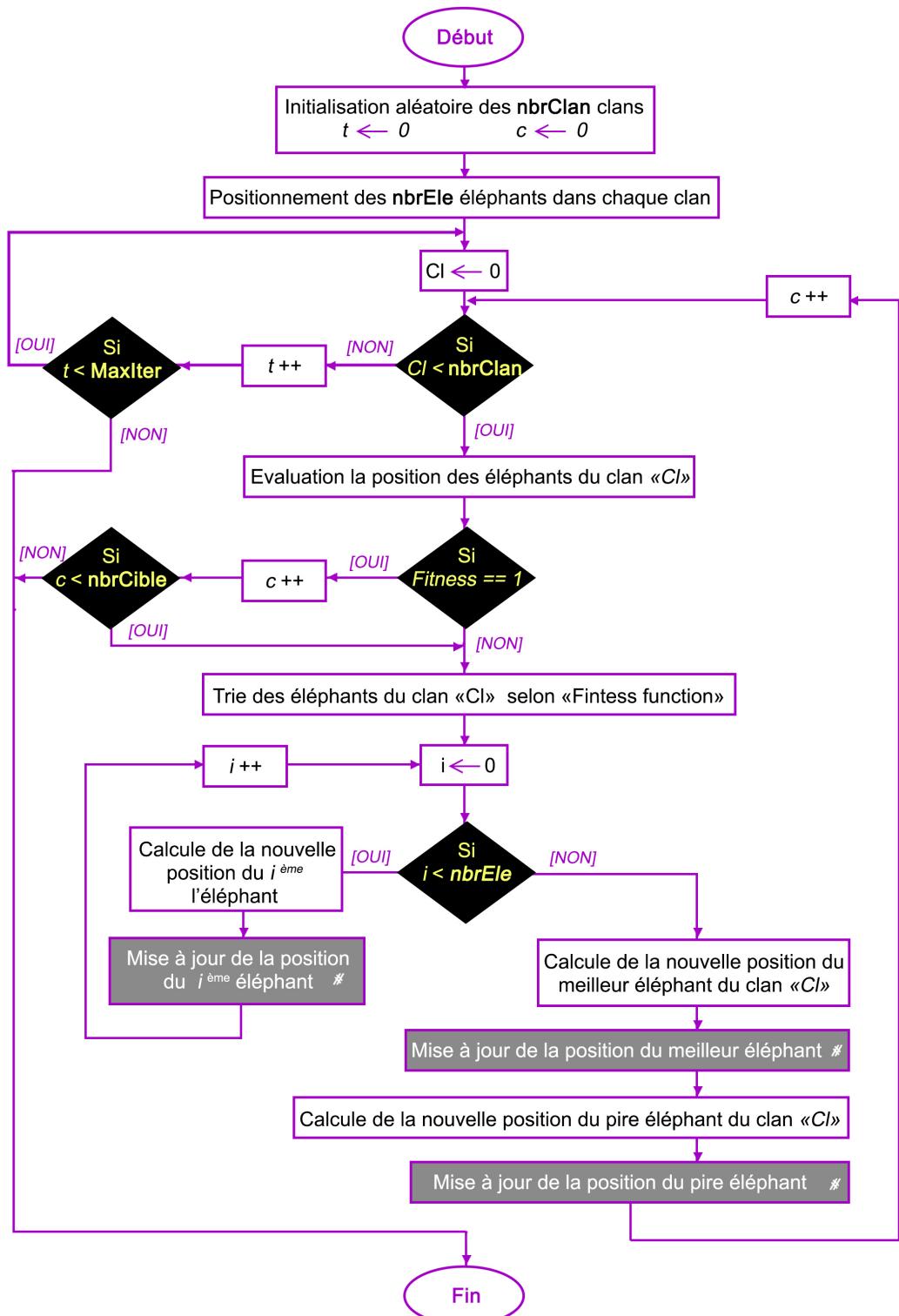
La génération de la prochaine position du pire éléphant de chaque clan grâce à l'opérateur de séparation de base aléatoire.

5.3.6 Déplacement des robots

Chaque robot éléphant " ele " se déplace vers la nouvelle position calculée via notre processus d'évitement d'obstacles.

5.3.7 Test de la condition d'arrêt

Incrémenter le nombre d'itérations effectuées " t ". Le processus de recherche prend fin si le nombre maximum d'itérations "**MaxIter**" est atteint. Dans le cas contraire, le processus reprend à partir de l'étape d'évaluation des solutions (section 5.3.3).



* Appel de la stratégie d'évitement d'obstacle avec l'ancienne et la nouvelle position calculée

Figure 5.5: Organigramme du mode de fonctionnement de l'approche EHO.

5.4 Adaptation de ESWSA pour le problème de la recherche de cibles

5.4.1 Solution

Tout comme EHO, une solution pour ESWSA est la position de coordonnées (x, y) occupée par un éléphant se comportant comme un robot.

5.4.2 Fonction objectif

La fonction objectif est la même que pour EHO, où les éléphants sont responsables de l'évaluation des solutions.

5.4.3 Éléphant

Un éléphant simule un robot, il occupe une case dans la grille de l'environnement. Il est doté d'une mémoire qui englobe sa meilleure position personnelle " $pbest$ " et la meilleure position globale " $gbest$ " du groupe d'éléphants. La dimension du vecteur de positions est alors égale à deux.

$$X_{i,2} = (x_{i1}, x_{i2}) \quad (5.5)$$

Pour des soucis de compréhension on a choisi la notation (x, y) avec i l'identifiant de l'éléphant

$$X_{i,2} = (x_i, y_i) \quad (5.6)$$

5.4.4 Outils de mémoire

"Pbest" dite "personal best", spécifique à chaque éléphant de l'essaim. Elle représente la position de l'éléphant dans l'environnement qui a maximisé la fonction objectif tout au long de la recherche. C'est la meilleure solution en qualité de l'éléphant.

$$\begin{aligned} Pbest_i &= (Pbest_{xi}, Pbest_{yi}) \\ Pbest_i^t &= \max_{1 \leq j \leq t} (f(X_i^j)) \end{aligned} \quad (5.7)$$

La mise à jour du $Pbest$ d'un éléphant au fil des itérations est représentée dans la figure 5.6 suivante :

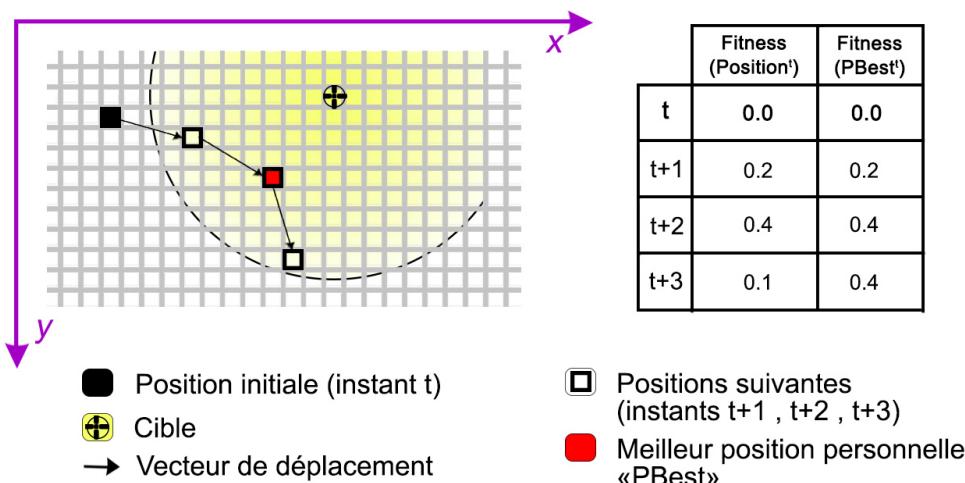


Figure 5.6: Représentation de la méthode de mise à jours du $Pbest$ d'un éléphant.

Global best () ou encore meilleure position globale, est une position spécifique à l'essaim, c'est la meilleure position trouvée par tout l'essaim. On peut dire que le *Gbest* consiste à trouver le max en termes de qualité des *Pbest* de chaque éléphant, comme le montre l'équation suivante :

$$\begin{aligned} Gbest &= (Gbest_x, Gbest_y) \\ Gbest^t &= \max(f(Pbest_i^t)) \end{aligned} \quad (5.8)$$

Chaque éléphant a connaissance de la valeur du *Gbest* à chaque itération, c'est une connaissance commune à tous.

5.4.5 Vélocité

La vélocité combine la vitesse et la direction qui détermine la prochaine position de l'éléphant. La notation de la vélocité dans un environnement bidimensionnel est la suivante :

$$V_{i,2} = (v_{xi}, v_{yi}) \quad (5.9)$$

5.4.6 Mécanisme de déplacement

5.4.6.1 Mise à jour de la vélocité de l'éléphant

La mise à jour de la vélocité est sujette à plusieurs paramètres dont les positions *Pbest* et *Gbest*. Comme on peut le voir dans la figure 5.7, d'après la constante *p* l'éléphant choisira de suivre sa meilleure position ou la meilleure du groupe, ce qui lui évite de stagner dans des optimums locaux.

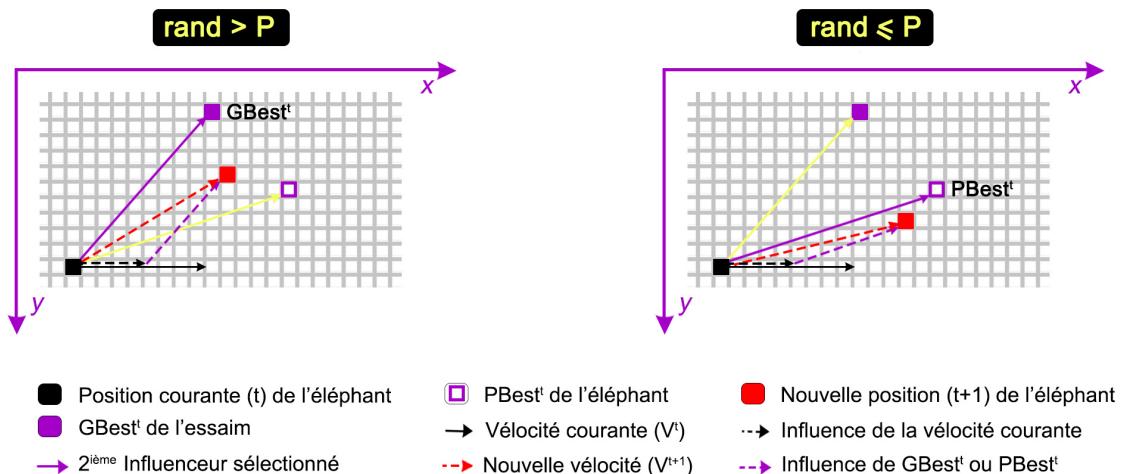


Figure 5.7: Représentation de la mise à jour de la vélocité d'un éléphant dans l'approche EWSA.

Vu qu'on se trouve dans un environnement bidimensionnel, on doit calculer deux vélocités, une selon les abscisses et une selon les ordonnées, ce processus est explicité dans les deux équations qui suivent:

$$v_{xi}^{t+1} = \begin{cases} v_{xi}^t * w^t + rand * (Gbest_x^t - x_i^t) & \text{si } random > p \\ v_{xi}^t * w^t + rand * (Pbest_{xi}^t - x_i^t) & \text{si } random \leq p \end{cases} \quad (5.10)$$

$$v_{yi}^{t+1} = \begin{cases} v_{yi}^t * w^t + rand * (Gbest_y^t - y_i^t) & \text{si } random > p \\ v_{yi}^t * w^t + rand * (Pbest_{yi}^t - y_i^t) & \text{si } random \leq p \end{cases} \quad (5.11)$$

Notre environnement a des bornes fixes, cette formule de calcul risque de générer des vélocités trop grandes ce qui perturbe l'organisation et le comportement du groupe d'éléphants. Une solution est de borner la vélocité. Soient les bornes de l'environnement représentées par le point $MaxP$

$$\begin{aligned} MaxP &= (MaxP_x, MaxP_y) \\ v_{xi}^{t+1} &= v_{xi}^{t+1} \bmod MaxP_x \\ v_{yi}^{t+1} &= v_{yi}^{t+1} \bmod MaxP_y \end{aligned} \quad (5.12)$$

5.4.6.2 Mise à jour de la position de l'éléphant

Tout comme la mise à jour de la position $X_{i,2}$ de l'éléphant i se fait sur deux niveaux celui des abscisses et celui des ordonnées, l'équation de mise à jour est la suivante :

$$\begin{aligned} x_i^{t+1} &= v_{xi}^{t+1} + x_i^t \\ y_i^{t+1} &= v_{yi}^{t+1} + y_i^t \end{aligned} \quad (5.13)$$

Toujours pour éviter les valeurs aberrantes, on procède de la même manière qu'avec la vélocité en bornant la position, soit :

$$\begin{aligned} MaxP &= (MaxP_x, MaxP_y) \\ x_i^{t+1} &= x_i^{t+1} \bmod MaxP_x \\ y_i^{t+1} &= y_i^{t+1} \bmod MaxP_y \end{aligned} \quad (5.14)$$

Ainsi, la position de l'éléphant est calculée en tenant compte de sa connaissance de sa position à l'instant t afin de calculer celle de l'instant $t + 1$.

5.4.6.3 Mise à jour de W^t (poids d'inertie)

Le poids d'inertie est un paramètre qui influence la vélocité et ainsi le calcul des positions des éléphants. Comme on a pu le voir dans le chapitre 2 section 3 page 19, l'équation choisie calcule le poids d'inertie selon le nombre d'itérations t avec l'initialisation suivante :

$$\begin{aligned} W_{max} &= 0.9 \\ W_{min} &= 0.1 \\ t_{max} &= 1000 \\ W^t &= W_{max} - \frac{W_{max} - W_{min}}{t_{max}} * t \end{aligned} \quad (5.15)$$

W^t joue un rôle important dans la convergence des éléphants vers la solution optimale, il détermine le taux de vélocité mémoire. C'est-à-dire la vélocité à l'instant t , qui influencera la prochaine vélocité à l'instant $t + 1$. Il permet alors de contrôler et varier la vitesse et la trajectoire afin d'explorer au mieux l'environnement de recherche.

5.5 Fonctionnement d'ESWSA

Dans ESWSA, chaque éléphant se comporte comme un robot indépendant du groupe mais coopératif. Doté d'une mémoire qui se présente en sa *pbest* et *gbest* qui sont nécessaires pour le choix des trajectoires possibles. L'organisation des éléphants lors de la recherche suit un schéma dont on explicite les étapes dans la figure 5.8 et dans le processus suivant :

5.5.1 Initialisation

Comme toute mét-heuristique, ESWSA commence par une étape d'initialisation déterminante pour le bon déroulement de la recherche à savoir :

5.5.1.1 Paramètres empiriques

Initialiser les paramètres empiriques consiste à leur choisir des valeurs fixes. Ils sont souvent le sujet d'un réglage pour augmenter l'efficacité et performance des approches basées essaim. Les paramètres d'ESWSA sont : T_{max} , $nbrEle$, P , W^t , $MaxP$, $MinP$.

5.5.1.2 Positions initiales des éléphants

L'emplacement initial des éléphants est aussi un facteur influent de la recherche. On initialise chaque éléphant à une position aléatoire en respectant les contraintes énoncées dans le chapitre 3.

5.5.1.3 Pbest pour chaque éléphant

Après avoir choisi les positions des éléphants, on doit évaluer la qualité de cette position. Cette évaluation permet d'initialiser les *Pbest* (Meilleure position personnelle en termes de qualité) de chaque éléphant de l'essaim.

5.5.1.4 Gbest

Comme on a pu l'expliquer plus haut dans 5.4.4, la position *Gbest* (Meilleure position globale de tout l'essaim) est initialisée selon le maximum des positions *pbest* évaluées en termes de qualité. En d'autres termes, celle qui détient la valeur maximum de la fonction objectif.

Une fois les initialisations faites, le processus d'ESWSA se met en marche selon ce qui suit :

5.5.2 Mise à jour de la vitesse pour chaque éléphant

Chaque éléphant de l'essaim se voit mettre à jour sa vitesse, cela se fait avec *le choix d'un nombre aléatoire* qui détermine l'équation à choisir. Si ce nombre aléatoire est inférieur au paramètre empirique *p* alors la vitesse sera calculée par rapport à sa position *Pbest*, sinon ça sera d'après le *Gbest*. Autrement dit l'éléphant choisira de privilégier le suivi de sa position *Pbest* ou bien *Gbest* à chaque itération.

5.5.3 Calcul de la prochaine position de chaque éléphant

Une fois la vitesse calculée, elle nous permet de mettre à jour la position de chaque éléphant selon l'équation de mise à jour voir section 5.4.6.2.

5.5.4 Déplacement des robots

Vu qu'un éléphant simule un robot, son déplacement se fait en employant la stratégie d'évitement d'obstacles, en cas d'environnement à obstacles.

5.5.5 Évaluation des positions des éléphants

Un éléphant a un angle de vue lui permettant de voir autour de lui sur une superficie de dix cases, ces cases seront évaluées selon la fonction objectif. L'éléphant gardera la position de la meilleure case.

Si une cible est détecté, le nombre de cibles trouvées " c " est alors incrémenté.

5.5.6 Mise à jour du P_{best} de chaque éléphant

Pendant la recherche, la position P_{best} est susceptible de changer, c'est pourquoi après chaque évaluation un test est effectué pour savoir si la mise à jour du P_{best} de l'éléphant est nécessaire. Ça signifie qu'on a trouvé une meilleure position nous rapprochons d'une cible.

5.5.7 Mise à jour de G_{best}

Après l'éventuelle mise à jour des positions P_{best} , vient la mise à jour de la position G_{best} , où un test est effectué également pour un potentiel changement du G_{best} .

5.5.8 Mise à jour du W^t (poids d'inertie)

Le poids d'inertie est relatif à chaque itération dont on calcule la valeur selon l'itération courante.

5.5.9 Critère d'arrêt de la recherche

Le processus d'ESWSA s'exécute en boucle, en incrémentant le nombre d'itérations " t ", jusqu'à la rencontre de toutes les "nbrCible" cibles ou bien jusqu'à atteindre le nombre maximum d'itérations "MaxIter".

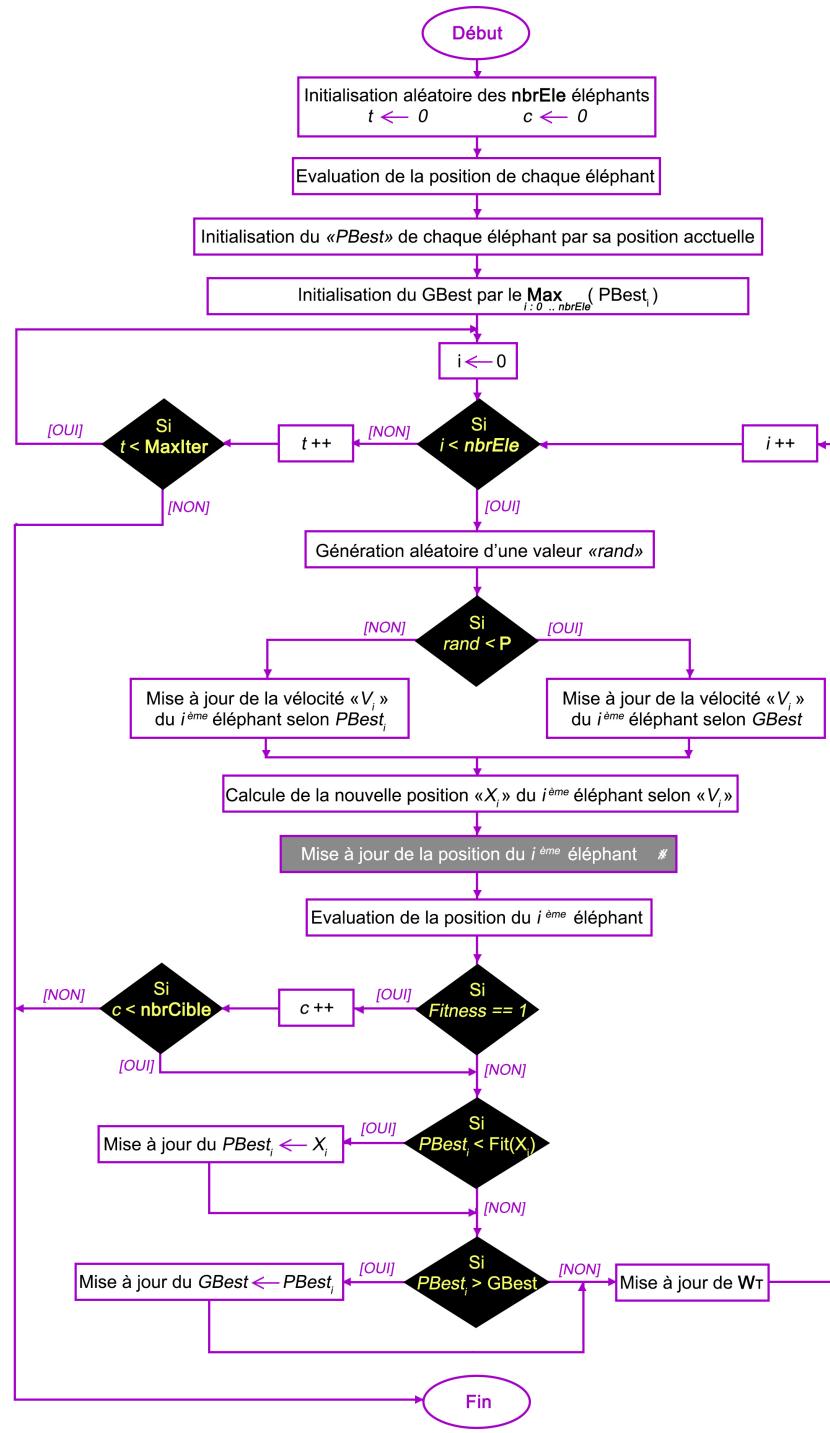


Figure 5.8: Organigramme du mode de fonctionnement de l'approche ESWSA

5.6 Conclusion

Les deux algorithmes présentés reposent sur le comportement des éléphants, mais vus sous deux angles bien différents. C'est pourquoi leur fonctionnement n'a rien en commun, exploitant chacun un aspect différent relatif aux éléphants et groupes d'éléphants. Le dernier chapitre sera consacré à la réalisation et expérimentations de toutes les approches citées dans ce chapitre et celui qui le précède.

Chapitre 6 : Validation Expérimentale

6.1 Introduction

Les expérimentations sont essentielles pour la validation des algorithmes et approches en général. Cette étape nécessite de déterminer les caractéristiques des machines utilisées ainsi que la description de la partie logicielle dont nous avons eu besoin, mais surtout une organisation cohérente des tests expérimentaux réalisés. C'est justement l'objet de ce chapitre, dont la structure est décrite comme suit :

- Description de l'environnement de développement.
- Expérimentations relatives au paramétrage des approches de recherche.
- Expérimentations sur les environnements, dont l'organisation est détaillée, suivies des résultats obtenus regroupés par types d'expérimentations et par types d'environnements.
- Présentation du simulateur accompagné de son manuel d'utilisation.

6.2 Environnement de développement

Comme pour toute expérimentation l'environnement de test est un élément clé pour la bonne interprétation des résultats obtenus.

6.2.1 Matériel

Nous avons utilisé deux machines (Laptop) possédant les caractéristiques et capacités suivantes :

Machine 1 :

Mémoire RAM: 8 Go

Processeur: Intel® Core™ i5-6200U
CPU 2.30GHz x 4

Carte graphique: Intel® HD
Graphics 520 (Skylake GT2)
x86/MMX/SSE2

Type de l'OS: UBUNTU 14.04 LTS 32-bit

Machine 2 :

Mémoire RAM: 8 Go

Processeur: Intel® Core™ i5-3317U @
1.70Hz 1.70GHz

Carte graphique: Intel®
HD Graphics 520

Type de l'OS: UBUNTU 18.04 LTS 64-bit

6.2.2 Logiciels

Le développement de nos algorithmes a été exclusivement fait sous système d'exploitation **Linux (Ubuntu)**, celui-ci muni des logiciels suivants:

IntelliJ IDEA : C'est un environnement de développement de logiciels informatiques orienté **Java**, supportant une large palette de plugins. C'est un des nombreux produits développés par la compagnie *JetBrains*. Celle-ci nous a été mise à disposition via une licence étudiant.

Nous l'avons utilisé pour le développement de notre solution de recherche de cibles.

Java : C'est un langage orienté objet académique qui a fait ses preuves, très largement utilisé pour ses nombreux avantages dont on cite: la portabilité, large diversité des librairies, conséquente documentation, ...etc.

Nous avons eu recours à une bibliothèque seulement, cela pour l'interface graphique en (**javaFx**) de notre simulateur en Java.

jfoenix-8.0.8.jar JFoenix¹ est une librairie mettant à disposition des composantes java qui implémentent Google Material Design, elle est open source et dédiée aux applications java, spécialement les interfaces *javaFx*.

PyCharm : C'est un environnement de développement de logiciels informatiques sous langage **Python**, aussi fournis par la compagnie *JetBrain*, nous avons obtenu ce logiciel sous licence étudiant. Nous l'avons principalement utilisé pour exploiter les résultats des expérimentations pour les traiter et traduire en graphes plus significatifs.

Python : C'est un langage de programmation interprété assez récent. Il est de plus en plus utilisé en vue de sa simplicité et polyvalence.

6.3 Expérimentations relatives aux approches

6.3.1 Paramétrage du mini-GA incrémental

Notre algorithme génétique possède quatre paramètres empiriques comme décrits dans le chapitre 3, nous avons procédé à une suite de tests afin de trouver les meilleures valeurs de ces paramètres, sans passer par un réglage de paramètres exhaustif. Pour cela, nous avons fixé les bornes de chaque paramètre en prenant en compte la taille d'une solution (3D ou 4D).

Les meilleurs paramètres obtenus se présentent comme suit :

- Taille de la population : 20
- Nombre d'itérations : 30
- Nombre de mutations : 2
- Point de croisement : aléatoire.

6.3.2 Paramétrage des approches de recherche de cibles

Pour chaque taille d'environnement, portée des cibles, nombre de cibles nous avons paramétré nos méthodes (BSO, Multi-BSO, EHO, EWSA) à l'aide de l'algorithme génétique (mini-GA incrémental), la moyenne des cinq meilleurs paramètres obtenus est calculée afin de former le meilleur paramétrage

¹Lien : <https://github.com/jfoenixadmin/JFoenix>

6.3.2.1 Réglage de Paramètres de BSO

Flip : Pour cette approche, nous avons constaté durant le réglage de ses paramètres que la variable *Flip* influence grandement l'évolution de la recherche. Une valeur de *Flip* trop grande empêche les abeilles d'explorer les zones distantes, de ce fait elle augmente les risques de stagnation, contrairement à des valeurs de *Flip* trop petites qui favorisent l'exploration en dispersant trop les abeilles, ce qui peut mener à une recherche désorganisée.

nbBees: Le nombre d'abeilles "*nbBees*" est un paramètre à double tranchant, car un grand nombre d'abeilles peut certes augmenter les chances de réussite, mais il amplifie considérablement le temps d'exécution.

maxChances pris avec une valeur trop grande cause une perte de temps et d'effort des abeilles, ce qui favorise une stagnation. En revanche, une valeur trop petite augmente les chances de passer à côté d'une solution intéressante.

Les meilleurs paramètres obtenus pour l'algorithme BSO sont présentés dans le tableau suivant :

Type	Portée des cibles						Taille d'environnement						Nbr cibles		
	mono-cible			multi-cibles			mono-cible			multi-cibles					
Paramètres	10	50	100	10	50	100	50	600	5000	50	600	5000	1	7	15
Flip	16	21	24	19	20	18	19	17	12	20	18	17	14	16	19
NbrBees	15	14	11	9	12	13	11	16	21	10	12	23	14	17	20
MaxChances	1	2	1	2	3	1	1	1	1	1	1	1	2	2	2

Table 6.1: Meilleurs paramètres pour BSO.

6.3.2.2 Réglage de Paramètres de Multi-BSO

Flip : Ce paramètre dans l'approche Multi-BSO ne doit pas être trop petit, car cela causerait une exploration trop grande et les abeilles se verront dispersées et perdraient leur organisation en groupes, d'autre part des valeurs exagérément grandes ralentiraient l'avancement de la recherche.

nbSwarms & nbBees & MaxChances : Cette méthode se distingue de la précédente par le nombre de groupes d'abeilles **nbSwarms**, dont les valeurs ont le même impact que le nombre d'abeilles **nbBees** dans BSO.

D'ailleurs le nombre d'abeilles **nbBees** par groupe et le paramètre **MaxChance** ont tous deux le même effet sur l'évolution de la recherche que sur BSO.

Le tableau ci-dessous résume les meilleurs paramètres trouvés pour l'approche Multi-BSO :

6.3.2.3 Réglage de Paramètres de EHO

nbClan & nbEle Le nombre de clans **nbClan** et d'éléphants par clan **nbEle** influencent directement les temps d'exécution et l'efficacité de la recherche, car des valeurs trop grandes (trop de clans et d'éléphants) augmentent les temps d'exécution, par contre leur affecter des valeurs trop petites réduit considérablement l'efficacité d'EHO.

Type	Portée des cibles						Taille d'environnement						Nbr cibles		
	mono-cible			multi-cibles			mono-cible			multi-cibles					
Paramètres	10	50	100	10	50	100	50	600	5000	50	600	5000	1	7	15
Flip	20	21	18	21	20	30	15	23	42	20	25	40	20	20	22
NbrBees	4	4	4	4	4	3	3	3	5	3	4	5	3	4	4
MaxChances	2	2	1	3	2	1	1	1	1	1	1	2	2	1	2
NbSwarms	3	4	4	4	4	3	3	4	5	4	4	5	4	4	4

Table 6.2: Meilleurs paramètres pour Multi-BSO.

Alpha (α) D'une part, si le paramètre α prend des valeurs trop grandes, les éléphants convergeront vers des positions proches de la meilleure solution trop vite, cela engendrera leur stagnation dans un minimum local. D'autre part, une valeur trop petite ralenti-rait le processus de recherche en raison des déplacements très petits entre deux itérations consécutives.

Beta (β) Enfin, une valeur trop proche du 1 pour β peut empêcher la convergence vers la solution optimale, car l'éléphant "matriarche" sera retenu par le centre de gravité du clan qui n'est pas forcément dans la direction de la meilleure solution. Par contre, avec une valeur trop petite (proche du 0) l'éléphant se détachera de son clan, ce qui causera de grands écarts dans les déplacements.

Les meilleurs paramètres de l'algorithme EHO sont donnés dans le tableau qui suit :

Type	Portée des cibles						Taille d'environnement						Nbr cibles		
	mono-cible			multi-cibles			mono-cible			multi-cibles					
Paramètres	10	50	100	10	50	100	50	600	5000	50	600	5000	1	7	15
nbrClan	3	3	2	4	4	4	3	4	4	4	4	5	3	4	4
nbrEle	3	3	3	3	3	3	3	3	4	3	4	4	3	3	4
alpha (α)	0.5	0.5	0.4	0.3	0.4	0.6	0.4	0.5	0.3	0.5	0.5	0.4	0.6	0.5	0.5
beta (β)	0.3	0.4	0.3	0.4	0.3	0.5	0.5	0.4	0.6	0.5	0.5	0.4	0.4	0.4	0.5

Table 6.3: Meilleurs paramètres pour EHO.

6.3.2.4 Réglage de Paramètres de EWSA

Nombre d'éléphants : Comme on a pu le supposer, l'augmentation du nombre d'éléphants a tendance à accélérer la recherche en termes de nombre d'itérations, mais aboutir à des solutions en des temps plus élevés.

Wt : Nous avons remarqué que quand le nombre d'itérations t augmente, la quantité du poids d'inertie W^t diminue. Autrement dit plus les éléphants avancent (nombre d'itérations) plus l'influence de l'ancienne vitesse diminue, ce qui permet une meilleure exploration.

Vélocité : L'initialisation de la vitesse influe sur la dynamique de la recherche, une initialisation aléatoire non-nulle $V_i = (valx, valy)$ à de grandes valeurs disperge les éléphants dans l'environnement. Contrairement à une initialisation nulle $V_i = (0, 0)$ qui permet des déplacements à pas progressifs.

L'initialisation à valeurs aléatoires bornées moyennes a été la plus satisfaisante, ces valeurs privilégient l'exploration au début de la recherche.

P : Lors du paramétrage à l'aide du GA, nous avons remarqué que lorsque p est très petit, chaque éléphant ne prend en considération que sa meilleure solution personnelle (P_{best_i}) ce qui ralentit le processus de recherche de cibles. Les éléphants auront tendance à repasser par les mêmes chemins et parfois stagner dans un optimum local, comme le montre les figures 6.1, 6.2 et 6.3 pour une valeur de $p = 0$.

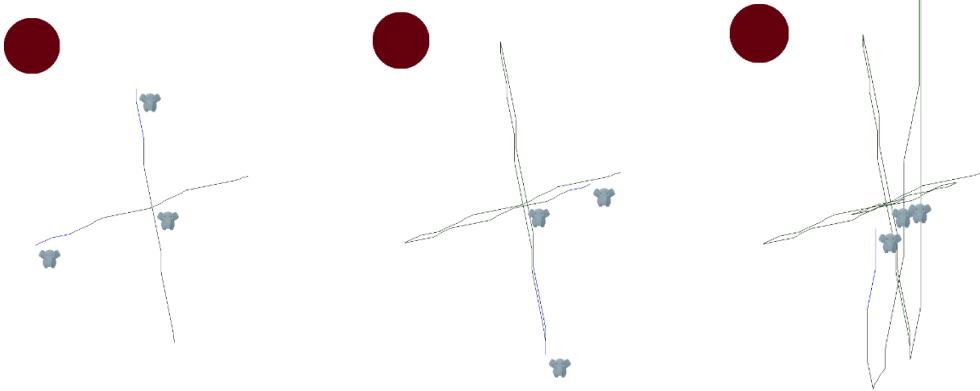


Figure 6.1: EWSWA à l'itération 4

Figure 6.2: EWSWA à l'itération 8

Figure 6.3: EWSWA à l'itération 16

Par contre, une grande valeur de p pousse les éléphants à suivre la même meilleure solution globale (G_{best}), convergeant ainsi vers une même cible ce qui cause un manque d'intensification.

Le tableau suivant illustre les meilleurs paramètres obtenus pour l'algorithme EWSWA :

Type	Portée des cibles						Taille d'environnement						Nbr cibles		
	mono-cible			multi-cibles			mono-cible			multi-cibles					
Paramètres	10	50	100	10	50	100	50	600	5000	50	600	5000	1	7	15
nbrEle	16	15	10	19	17	14	12	14	19	16	16	18	17	15	20
Wt	0.4	0.5	0.4	0.4	0.5	0.3	0.6	0.5	0.5	0.6	0.5	0.5	0.5	0.4	0.5
p	0.6	0.5	0.3	0.5	0.5	0.5	0.5	0.5	0.3	0.6	0.5	0.5	0.5	0.5	0.5

Table 6.4: Meilleurs paramètres pour EWSWA.

6.4 Expérimentations relatives à l'environnement

Nous avons sélectionné trois paramètres à expérimenter pour observer le comportement de nos approches de recherche basées essaims. Ces paramètres sont : *la portée des cibles*, *la taille de l'environnement* et *le nombre de cibles recherchées*.

Pour cela, nous avons réalisé ces expérimentations pour les trois types d'environnements précédemment explicités dans le chapitre 3, qui sont : **simples**, **avec obstacles** et **complexes**.

6.4.1 Organisation des types d'expérimentations

Pour une question de clarté, nous avons jugé nécessaire de décrire comment est organisé chaque type d'expérimentation avant de passer aux résultats obtenus.

6.4.1.1 Organisation des expérimentations par rapport à la portée des cibles

Pour les expérimentations relatives à l'influence de la portée des cibles sur le comportement de nos algorithmes de recherche, nous avons fixé la taille de l'environnement à 500

$\times 500$ positions en faisant varier la portée d'un rayon de 10 positions à 100 positions avec un pas de 10.

Ainsi comme le montre la figure 6.4, les tests de ce type proviennent des exécutions sur 40 environnements (4 par portée), ce qui fait 400 exécutions pour le mono-cible et 400 autres pour le multi-cibles, pour chaque méta-heuristique (après paramétrage).

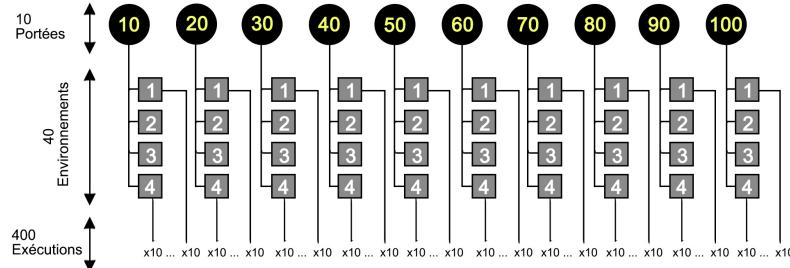


Figure 6.4: Représentation de l'organisation des expérimentations sur la portée des cibles.

6.4.1.2 Organisation des expérimentations par rapport à la taille de l'environnement

Pour les expérimentations sur l'influence de la taille de l'environnement sur le comportement de nos méta-heuristiques à la recherche de cibles, nous avons adapté la portée des cibles à chaque taille d'environnement conformément à l'équation suivante :

$$portee = \frac{taille_{Coté} \times 10}{100} = \frac{taille_{Coté}}{10} \quad (6.1)$$

Ce qui revient à :

$$S_{cible} = \frac{\sqrt{S_{env}}}{5} \times \pi \quad (6.2)$$

Avec :

- $taille_{Coté}$: taille du côté de l'environnement carré.
- S_{cible} : surface d'émission de la cible (rayon = portée).
- S_{env} : surface de l'environnement de recherche ($S_{env} = taille_{Coté}$).

Pour cela nous avons sélectionné dix différentes tailles d'environnement, les tailles du coté de ces environnements sont comme suit : 50, 100, 200, 400, 600, 800, 1000, 1500, 3000, 5000.

Les surfaces respectives sont les suivantes: 2500, 10000, 40000, 160000, 360000, 640000, 1000000, 2250000, 9000000, 25000000.

Les portées correspondantes sont: 5, 10, 20, 40, 60, 80, 100, 150, 300, 500.

Les résultats des tests liés à ce type d'expérimentation sont le résultat de 40 exécutions par taille d'environnement (4 configurations pour chaque taille), ce qui revient à un total de 400 exécutions pour le mono-cible et 400 autres pour le multi-cibles et ce pour chaque approche (après paramétrage). Le schéma 6.5 ci-dessous illustre cette organisation.

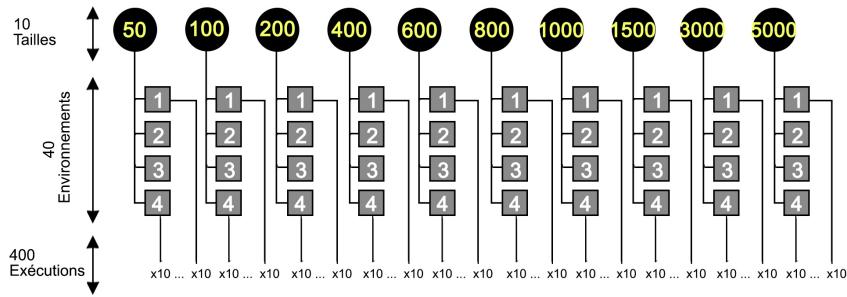


Figure 6.5: Représentation de l'organisation des expérimentations sur la taille des environnements.

6.4.1.3 Organisation des expérimentations par rapport au nombre de cibles

Ces expérimentations permettent d'étudier l'impact du changement du nombre de cibles présentes dans nos environnements sur le comportement de nos méta-heuristiques (BSO, EHO, EWSA et Multi-BSO). Pour cela, nous avons fixé la taille des environnements à 500 × 500 positions ainsi qu'une portée de cible égale à 50 (Conformément à l'équation 6.1).

Nous avons pris le nombre de cibles compris entre 1 et 15 (bornes incluses) avec un pas de 2, ce qui nous donne les 8 nombres de cibles suivants : 1, 3, 5, 7, 9, 11, 13, 15.

De ce fait, les résultats des tests sont le fruit d'exécutions sur 32 environnements (4 par nombre de cibles), ce qui correspond à 320 exécutions pour chacune des approches développées (après paramétrage).

La figure 6.6 ci-dessous schématisse cette organisation :

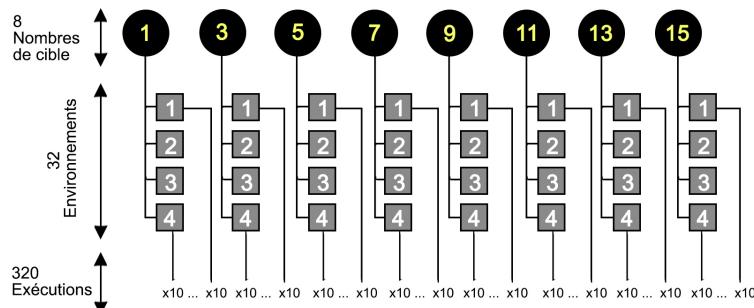


Figure 6.6: Représentation de l'organisation des expérimentations sur le nombre de cibles.

6.4.2 Expérimentations par rapport à la portée des cibles

6.4.2.1 Taux de réussite

La figure 6.7 est un *diagramme à bandes* représentant l'évolution du taux de réussite de chacune de nos méta-heuristiques par rapport aux différentes portées des cibles.

Nous constatons que le taux de réussite est maximal (100%) quelle que soit la valeur de la portée (de 10 à 100), autrement dit les algorithmes implémentés arrivent toujours à trouver la ou les cible(s) sans atteindre le nombre maximal d'itérations qui est de 1000.

Il est à noter que ces mêmes résultats ont été obtenus pour les trois types d'environnements (simple, avec obstacles et complexe) de dimensions 500×500 aussi bien dans les cas de mono-cible que de multi-cibles.

6.4.2.2 Variation du nombre d'itérations et temps d'exécution

a- Environnements simples (sans obstacles)

- Mono-cible

Les figures 6.8 et 6.9 illustrent des *diagrammes à lignes brisées*, représentant respectivement la variation du nombre d'itérations et temps d'exécution de chacune de nos approches confrontées aux différentes portées de la cible dans un environnement sans obstacle.

On remarque que toutes les méthodes voient leur nombre d'itérations décroître avec l'élargissement de la portée de la cible. Pour les méthodes Multi-BSO, EHO et EWSA, le nombre d'itérations est estimé entre 12 et 16 pour une portée égale à 10, puis décroît progressivement jusqu'à atteindre environ 2 ou 3 itérations pour la portée maximale. Quant à BSO, il débute avec un peu plus d'itérations (moyenne de 33), avant de diminuer pour rejoindre les autres méthodes lorsque la portée est maximale.

Pour ce qui est des temps d'exécution, nos quatre méthodes s'exécutent en des temps records de moins de 0.08 secondes. BSO et Multi-BSO possèdent les meilleurs temps relativement stables et inférieurs à 0.02 secondes, suivies d'EHO atteignant les 0.05 secondes, enfin EWSA est l'approche la plus lente en arrivant jusqu'à 0.08 secondes pour une portée égale à 100.

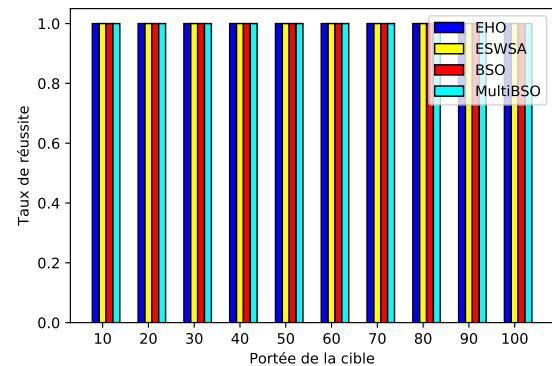


Figure 6.7: Comparaison de la variation du taux de réussite des algorithmes selon la portée des cibles.

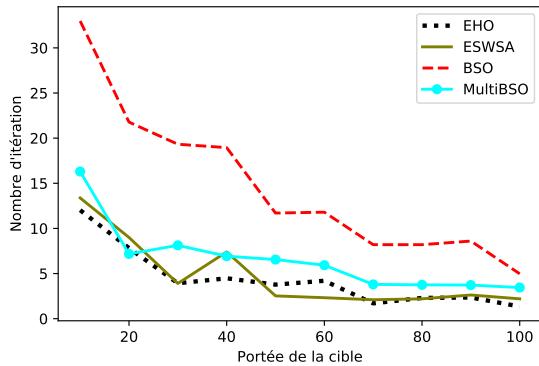


Figure 6.8: Comparaison de la variation du nombre d’itérations des algorithmes selon la portée de la cible (sans obstacles).

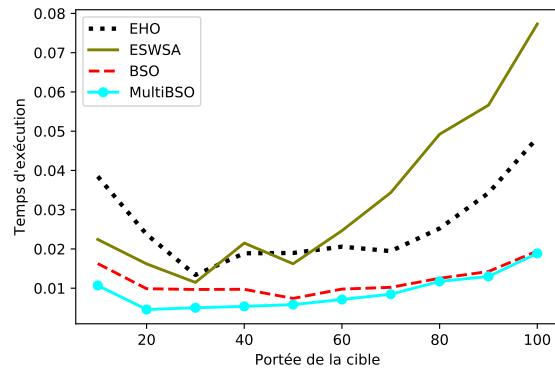


Figure 6.9: Comparaison de la variation du temps d’exécution des algorithmes selon la portée de la cible (sans obstacles).

- Multi-cibles

Les deux figures ci-dessous 6.10 et 6.11 sont représentatives de la variation du nombre d’itérations et temps d’exécution (dans cet ordre) de nos algorithmes, faces aux différentes portées des cibles dans des environnements sans obstacles.

Nous distinguons trois comportements par rapport au nombre d’itérations, le premier est propre à BSO, celui-ci détient le plus grand nombre d’itérations pour toutes les portées, malgré la réduction de ce nombre de 107 à 29 en vue de l’augmentation des valeurs de la portée des cibles. Compte au 2^{ème}, il est caractérisé par un nombre d’itérations avoisinant les 63 itérations pour une portée égale à 10, puis une diminution de ce nombre jusqu’à atteindre 8 à 4 itérations pour les cibles de portée 100; ce comportement concerne les deux mét-heuristiques EHO et ESWSA. Enfin, le 3^{ème} comportement est celui de Multi-BSO, possédant le plus petit nombre d’itérations initial 31 qui diminue graduellement jusqu’à 9 itérations pour la portée égale à 100s.

En termes de temps d’exécution, seul ESWSA subit une augmentation visible allant de 0.15 à 0.83 secondes pour les portées de 10 à 100 respectivement, les trois autres algorithmes ne dépassent pas les 0.23 secondes, cette valeur correspondant aux résultats d’EHO pour la portée de 100 positions, BSO et Multi-BSO ayant des temps légèrement inférieurs.

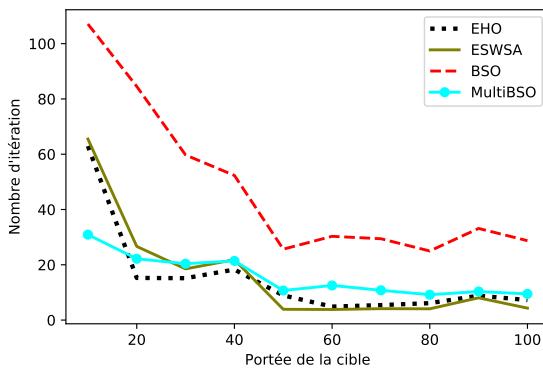


Figure 6.10: Comparaison de la variation du nombre d’itérations des algorithmes selon la portée des cibles (sans obstacles).

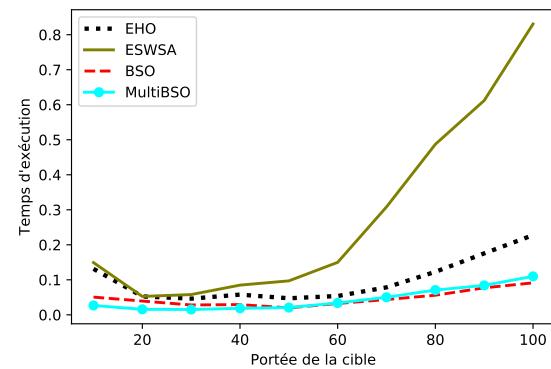


Figure 6.11: Comparaison de la variation du temps d’exécution des algorithmes selon la portée des cibles (sans obstacles).

b- Environnements avec obstacles

- Mono-cible

Les deux *diagrammes à lignes brisées* des figures 6.12 et 6.13 retracent la variation du nombre moyen d’itérations et temps moyens d’exécution de nos algorithmes faisant face à une portée croissante de la cible dans des environnements avec obstacles.

Pour la première valeur de portée (égale à 10) l’approche BSO est celle qui nécessite le plus d’itérations (52.68), suivie d’EHO et Multi-BSO dont le nombre d’itérations est d’environ 14, enfin ESWSA, qui démarre avec seulement 3.58 itérations.

Pour toutes les mét-heuristiques, ce nombre évolue de manière inversement proportionnelle à la portée de la cible jusqu’à ce que la portée soit égale à 50 où il commence à se stabiliser autour de 9 itérations.

Les temps d’exécution sont minorés par 0.001 secondes et majorés par la valeur 0.11 secondes, ce qui est remarquablement rapide. On distingue une augmentation des temps d’exécution pour ESWSA, plus légère pour EHO suite à l’élargissement de la surface de la portée de la cible. Les deux autres approches restent moyennement stables n’excédant pas les 0.02 secondes.

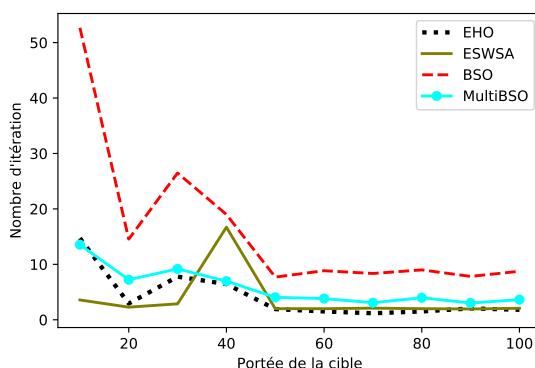


Figure 6.12: Comparaison de la variation du nombre d’itérations des algorithmes selon la portée de la cible (avec obstacles).

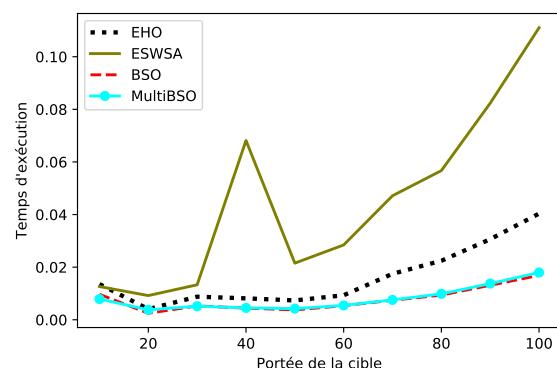


Figure 6.13: Comparaison de la variation du temps d’exécution des algorithmes selon la portée de la cible (avec obstacles).

- Multi-cibles

Les figures 6.14 et 6.15 reflètent la variation du nombre d'itérations et temps d'exécution de nos quatre approches de recherche face aux portées des cibles, cela dans des environnements avec obstacles.

Le nombre d'itérations diminue avec l'augmentation de la portée des cibles pour toutes les approches, néanmoins BSO se distingue par une diminution importante en passant d'une moyenne de 110.6 à 22.58 itérations. Contrairement à Multi-BSO, EHO et EWSWA qui varient entre 30 et 3 itérations.

Pour ce qui est des temps d'exécution, nous remarquons que pour EWSWA la durée de recherche croît considérablement avec l'augmentation de la portée des cibles, comme nous pouvons le voir, il passe de 0.06 à 2.73 secondes. Les autres algorithmes qui se comportent de manière similaire, mais avec une marge moins importante passant de 0.01 à 0.23 secondes.

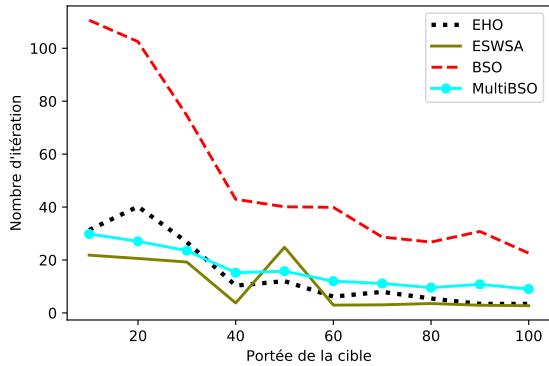


Figure 6.14: Comparaison de la variation du nombre d'itérations des algorithmes selon la portée des cibles (avec obstacles).

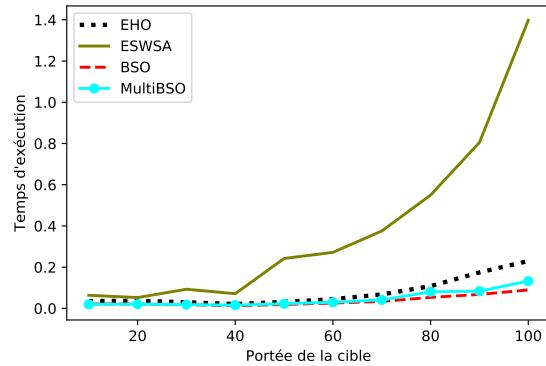


Figure 6.15: Comparaison de la variation du temps d'exécution des algorithmes selon la portée des cibles. (avec obstacles)

c- Environnements complexes

- Mono-cible

Les *diagrammes à lignes brisées* des figures 6.16 et 6.17, décrivent la variation du nombre d'itérations et temps d'exécution de nos approches, celles-ci confrontées aux diverses valeurs de portée de la cible recherchée et cela, dans des environnements complexes.

Nous remarquons que le nombre d'itérations décroît avec l'augmentation de la portée de la cible, BSO débute avec le plus grand nombre de 60.83 pour en effectuer que 6.60 itérations lorsque la portée est à 100. EWSWA passe de 40.08 à 2.03 itérations. Quant à EHO, lui fluctue entre 22.13 et 1.48 itérations. Enfin Multi-BSO possède la courbe descendante la plus régulière qui passe de 25 à 3.05 itérations.

Pour ce qui est des temps d'exécution, ceux de BSO, Multi-BSO ainsi qu'EOH connaissent une légère augmentation de 0.02 à 0.04 secondes contrairement à ceux d'EWSWA qui double passant de 0.06 à 0.12 secondes.

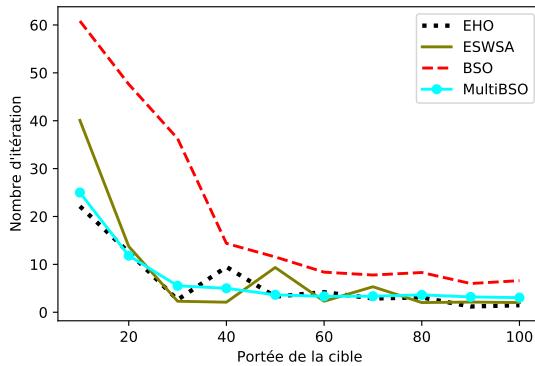


Figure 6.16: Comparaison de la variation du nombre d’itérations des algorithmes selon la portée de la cible (complexe).

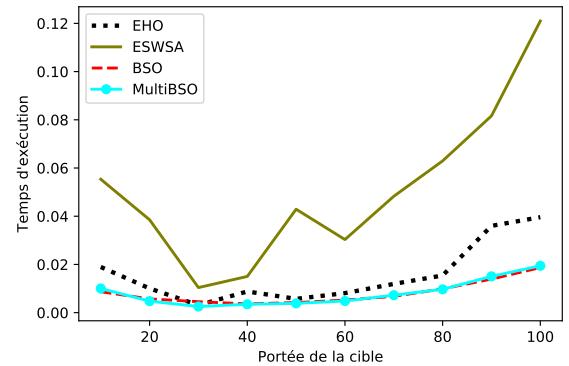


Figure 6.17: Comparaison de la variation du temps d’exécution des algorithmes selon la portée de la cible (complexe).

- Multi-cibles

À travers les figures 6.18 et 6.19, sont représentées respectivement la variation du nombre d’itérations et temps d’exécution de nos approches sous-mises aux dix valeurs de portée des cibles dans des environnements complexes.

Comme dans le mode mono-cible, l’augmentation de la portée des cibles conduit à une baisse du nombre d’itérations. On note pour la mét-heuristique BSO une importante chute du nombre d’itérations allant de 110.6 à 22.58 suivie d’EHO et Multi-BSO qui se comportent de façon similaire avec un nombre d’itérations initial aux alentours de 80 qui dégringole aux environs de 8 itérations. Enfin, ESWSA possède la baisse la moins flagrante passant de 46.88 à 3.08 itérations atteignant le nombre minimum d’itérations qui est de 3.

Une hausse du taux de croissance est observée pour les temps d’exécution d’ESWSA (de 0.05 à 1.10 secondes). Contrairement à ceux de BSO, Multi-BSO et EHO où une stabilisation demeure avec une très légère augmentation pour EHO ne dépassant pas les 0.21 secondes, pour les portées entre 60 et 100.

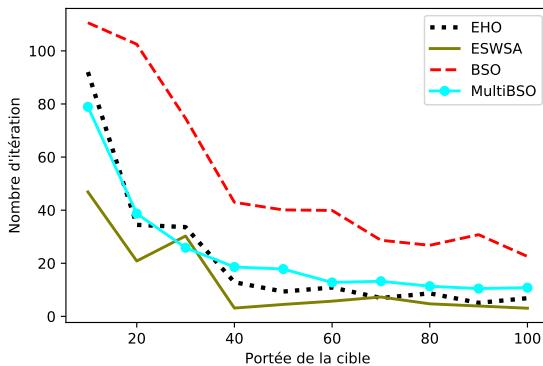


Figure 6.18: Comparaison de la variation du nombre d’itérations des algorithmes selon la portée des cibles (complexe).

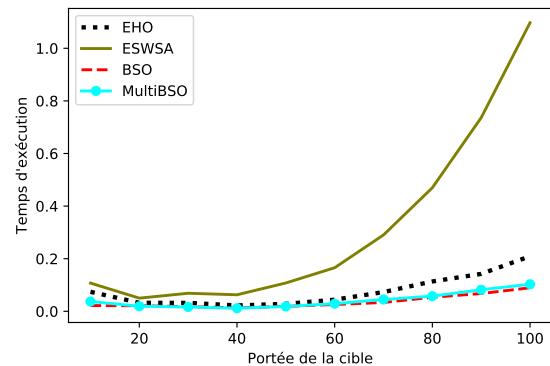


Figure 6.19: Comparaison de la variation du temps d’exécution des algorithmes selon la portée des cibles (complexe).

6.4.2.3 - Analyse relative à la portée des cibles

À travers les résultats commentés des cas du mono-cible et multi-cibles appliqués à toutes les approches, on peut retenir les points suivants :

- Même pour de petites portées des cibles, nos algorithmes arrivent à bout de la recherche avec succès.
- L'élargissement graduel de la portée des cibles a fait fléchir le nombre d'itérations.
- Malgré les temps d'exécution acceptables, ils sont moyennement plus importants pour le mode multi-cibles.
- La densité des obstacles influe la recherche ce qui s'explique par l'augmentation des temps d'exécution et nombre d'itérations.

6.4.3 Expérimentations par rapport à la taille de l'environnement

6.4.3.1 Taux de réussite

a- Environnements simples (sans obstacles)

- Mono-cible

Le graphe de la figure 6.20 montre la variation du taux de réussite pour chaque approche développée, celles-ci sous-mises à diverses tailles d'environnements sans obstacles.

Pour EHO et EWSA, un taux maximal de réussite est observé quelle que soit la taille de l'environnement. En revanche, BSO et Multi-BSO voient leur taux de succès maximal diminuer jusqu'à 60% pour BSO et 77% pour Multi-BSO pour les deux dernières tailles c'est-à-dire 3000 et 5000.

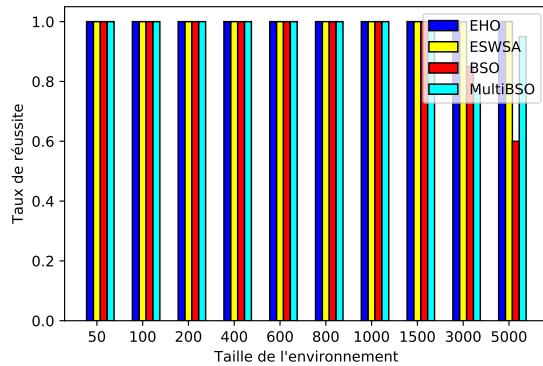


Figure 6.20: Comparaison de la variation du taux de réussite des algorithmes selon la taille de l'environnement (sans obstacles).

- Multi-cibles

La figure 6.21 à droite décrit les taux de réussite dans la recherche des cibles, de chaque mét-heuristique en fonction de la variation de la taille des environnements sans obstacles.

Nous nous apercevons que comme pour le mono-cible, EHO et ESWSA atteignent le taux de 100% pour toutes les tailles, ce qui n'est pas le cas de BSO dont le taux de réussite décroît à partir de la 7^{ème} taille d'environnement avec 99% pour atteindre 51% pour les environnements de taille 5000 et Multi-BSO dont les taux de succès des deux dernières tailles d'environnement sont respectivement de 85% et 79%.

b- Environnements avec obstacles

- Mono-cible

La figure 6.22 affiche la variation des taux de réussite de nos approches selon des tailles croissantes des environnements avec obstacles.

Pour les tailles entre 50 et 1 500, tous nos algorithmes arrivent à obtenir les 100% de taux de succès, toutefois les environnements de dimensions 3 000 connaissent une baisse de ce taux à 76% pour BSO, mais toujours 100% pour les autres, enfin pour les plus grands environnements Multi-BSO atteint un taux de 87% contre 55% pour BSO et 100% pour EHO et ESWSA.

- Multi-cibles

La figure 6.23 représente la variation des taux de succès de nos approches confrontées à plusieurs tailles d'environnement avec obstacles.

EHO et ESWSA arrivent à maintenir les 100% de taux de réussite pour toutes les tailles testées, contrairement à Multi-BSO qui relâche son taux de réussite à 83% au bout de l'avant-dernière taille (3 000), et BSO qui déjà à partir des tailles 1 000 baisse à 67% pour continuer à baisser jusqu'à 17%. Toujours dans la limite du nombre d'itérations maximal (1 000).

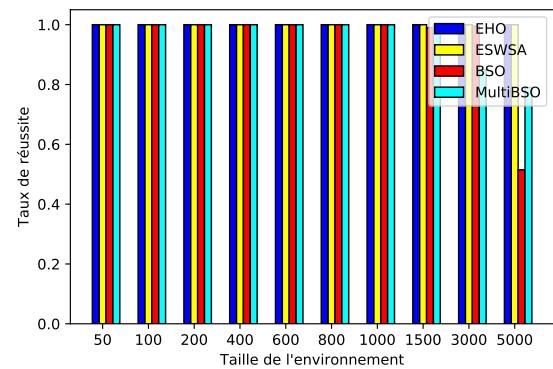


Figure 6.21: Comparaison de la variation du taux de réussite des algorithmes selon la taille de l'environnement (sans obstacles).

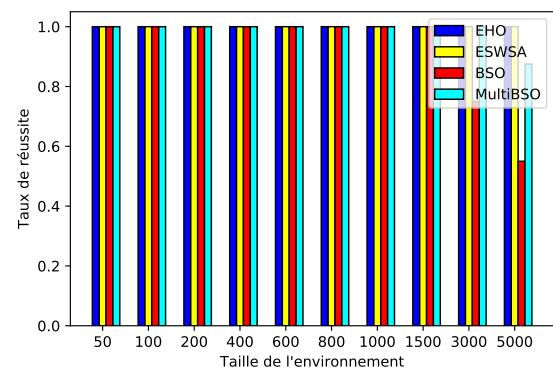


Figure 6.22: Comparaison de la variation du taux de réussite des algorithmes selon la taille de l'environnement (avec obstacles).

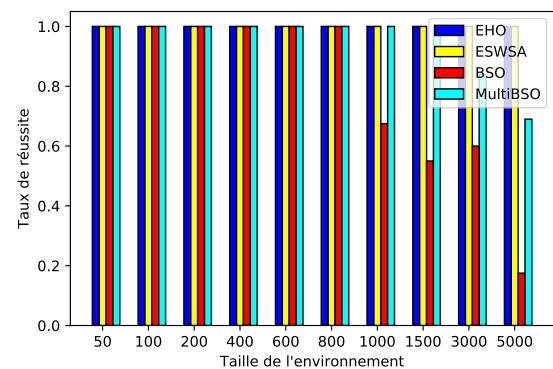


Figure 6.23: Comparaison de la variation du taux de réussite des algorithmes selon la taille de l'environnement (avec obstacles).

c- Environnements complexes

- Mono-cible

Dans la figure 6.24 est représentée la variation des taux de réussite de l'ensemble de nos approches exécutées sur différentes tailles d'environnements complexes.

Pour toutes les tailles testées de 50 à 3 000 la totalité de nos algorithmes atteignent les 100% de réussite, mais pour la dernière taille d'environnement (5 000) seuls EHO et EWSA arrivent à garder le même taux de réussite, tels que Multi-BSO décent à 90% et BSO chute à 60%, car ils sont limités par le nombre d'itérations maximal.

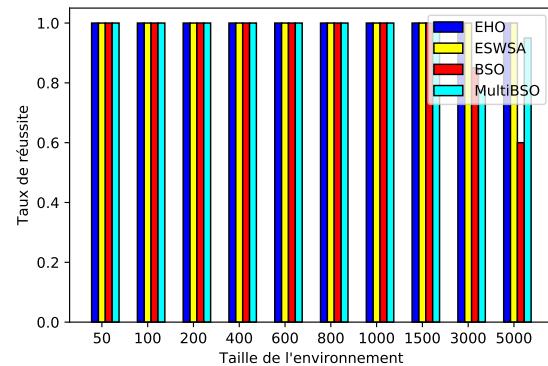


Figure 6.24: Comparaison de la variation du taux de réussite des algorithmes selon la taille de l'environnement (complexe).

- Multi-cibles

Les *diagrammes à bandes* de la figure 6.25 décrivent comment les taux de succès de nos algorithmes varient faces aux tailles d'environnements complexes à la recherche de plusieurs cibles.

EHO et EWSA maintiennent les 100% de taux de réussite pour toutes les tailles d'environnement, idem pour Multi-BSO sauf pour ce qui est de la dernière taille, il n'arrive qu'à 71%. Par contre BSO décent sous la barre des 100% dès la taille 1000, à partir de là, son taux fluctue entre 36% et 98% car freiné par le nombre d'itérations maximal.

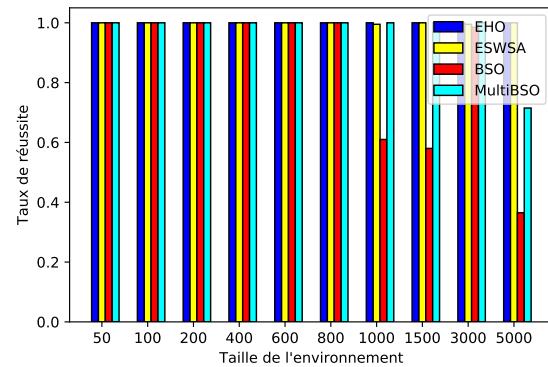


Figure 6.25: Comparaison de la variation du taux de réussite des algorithmes selon la taille de l'environnement (complexe).

6.4.3.2 Variation du nombre d'itérations et temps d'exécution

a- Environnements simples (sans obstacles)

- Mono-cible

Dans les figures 6.26 et 6.27 sont représentés la variation du nombre d'itérations et du temps d'exécution de nos méta-heuristiques pour l'ensemble des tailles d'environnement sans obstacles sélectionnées.

Les méta-heuristiques inspirées des éléphants détiennent le nombre d'itérations le plus réduit, celui-ci varie entre 1 et 10 itérations pour EHO et entre 2 et 13 itérations pour EWSA. Par contre, les deux variantes de BSO connaissent une importante augmentation du nombre d'itérations de 4 à 574 itérations pour BSO et de 2 à 189 pour Multi-BSO, suite à l'augmentation de la taille des environnements.

Pour ce qui est de l'aspect "temps", toutes les approches sont sujettes à une croissance proportionnelle à la croissance des tailles d'environnement à différents coefficients près, tels que nous pouvons les ordonner selon la vitesse d'augmentation des temps

d'exécution de l'algorithme le plus lent au plus rapide comme suit : ESWSA puis BSO suivis de Multi-BSO et enfin EHO, ce dernier atteint un maximum de 4.6 secondes pour une taille = 5 000.

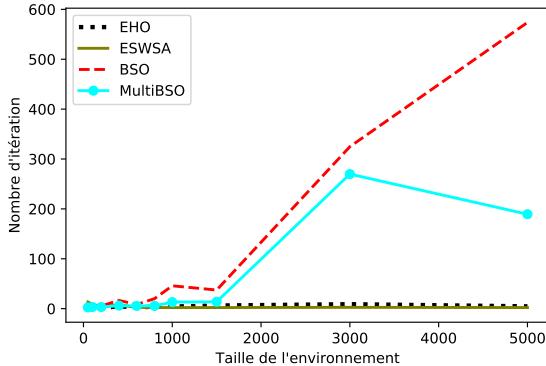


Figure 6.26: Comparaison de la variation du nombre d'itérations des algorithmes selon la taille de l'environnement (sans obstacles).

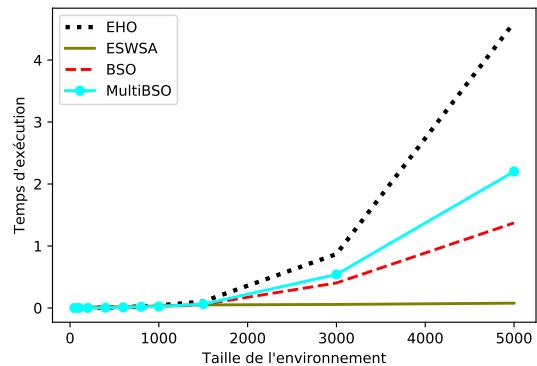


Figure 6.27: Comparaison de la variation du temps d'exécution des algorithmes selon la taille de l'environnement (sans obstacles).

- Multi-cibles

Les courbes des figures 6.28 et 6.29 illustrent la variation du nombre moyen d'itérations et temps moyens d'exécution de nos algorithmes vis-à-vis des différentes tailles d'environnement sans obstacles.

Ici aussi, pour le mode multi-cibles EHO et ESWSA effectuent le moins d'itérations, avec une légère augmentation allant de 2.43 jusqu'à 19.43 itérations pour EHO, de 3.73 à 6.95 itérations pour ESWSA, cela pour l'agrandissement de la taille des environnements. Par ailleurs, BSO et Multi-BSO subissent une conséquente élévation du nombre d'itérations allant de 9.88 à 991.18 itérations pour BSO et de 5.95 à 865.53 pour ce qui est de Multi-BSO.

Quant aux temps d'exécution, globalement nous pouvons dire que les temps d'exécution des algorithmes croient avec l'augmentation de la taille des environnements à différentes vitesses, tels que BSO passe progressivement de 0.001 à 6.57 secondes suivi de Multi-BSO avec des temps haussant de 0.001 à 11.75 secondes, puis EHO qui démarre à 0.001 pour atteindre les 19.43 secondes et enfin ESWSA qui enregistre les plus grands temps passant de 0.001 à 17.01 secondes.

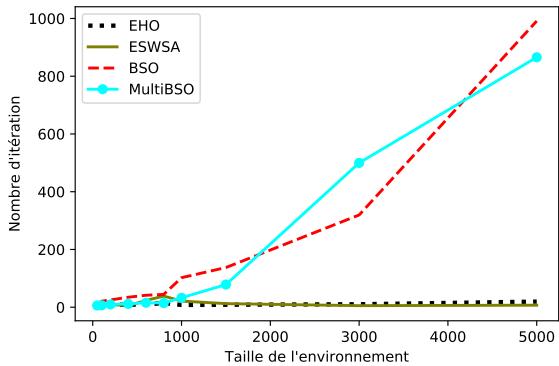


Figure 6.28: Comparaison de la variation du nombre d'itérations des algorithmes selon la taille de l'environnement (sans obstacles).

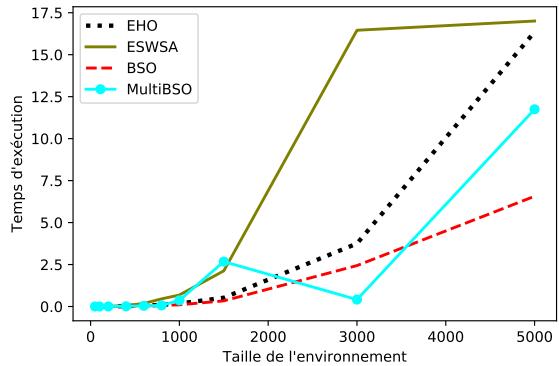


Figure 6.29: Comparaison de la variation du temps d'exécution des algorithmes selon la taille de l'environnement (sans obstacles).

b- Environnements avec obstacles

- Mono-cible

Les *diagrammes à lignes brisées* des figures 6.30 et 6.31 font apparaître la variation du nombre d'itérations et temps d'exécution de BSO, Multi-BSO, EHO ainsi qu'ESWSA par rapport aux tailles des environnements avec obstacles.

La figure de gauche montre que BSO détient le plus grand nombre d'itérations pour l'ensemble des tailles d'environnement, avec une augmentation allant de 2.55 à 566.18 itérations. Multi-BSO vient juste après avec des nombres d'itérations croissants tout en restant réduit jusqu'aux environnements de 3000×3000 où il arrive à 48.45 itérations qui par la suite grimpe vers 295.58 itérations pour les plus grands environnements. Enfin, EHO et ESWSA disposent des nombres d'itérations les plus réduits, fluctuant entre 1.08 et 29.78 itérations.

Quant à la figure de droite, nous y discernons deux comportements tous deux croissants avec l'élargissement des surfaces des environnements, le 1^{er} englobe BSO, Multi-BSO et EHO qui possèdent des temps d'exécution minimes (entre 0.001 et 3.83 secondes), et le 2^{ème} comportement concerne ESWSA avec des temps moyens plus ou moins élevés, allant de 0.001 à 15.56 secondes.

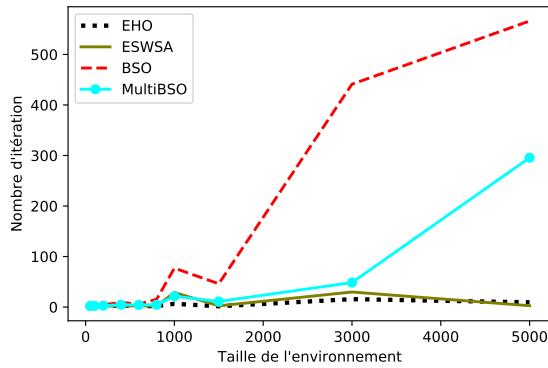


Figure 6.30: Comparaison de la variation du nombre d'itérations des algorithmes selon la taille de l'environnement (avec obstacles).

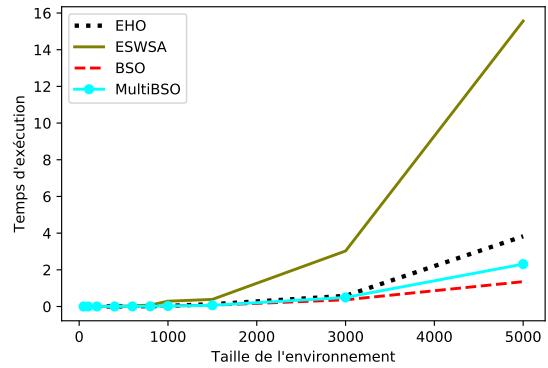


Figure 6.31: Comparaison de la variation du temps d'exécution des algorithmes selon la taille de l'environnement (avec obstacles).

- Multi-cibles

Les figures 6.32 et 6.33 présentées ci-dessous, retracent la variation du nombre d'itérations et temps d'exécution de nos algorithmes testés sur plusieurs tailles d'environnement avec obstacles à la recherche de plusieurs cibles.

BSO et Multi-BSO se distinguent avec des nombres élevés d'itérations, BSO débutant de 9.50 itérations et arrivant à la limite maximale de 1000 itérations, quant à Multi-BSO il passe d'une moyenne de 5.40 à 920.68 itérations. D'autre part EHO et ESWSA possèdent le moins d'itérations (entre 2.45 et 22.18 itérations).

Côté temps d'exécution, nous pouvons ordonner nos méthodes de la plus rapide à la plus long comme suit : BSO avec de 0.001 à 1.99 secondes suivie de Multi-BSO avec de 0.01 à 5.67 secondes puis EHO avec des temps entre 2.45 et 20.50 secondes, pour finir ESWSA dont les temps croissent de manière spectaculaire de 0.001 à 94.55 secondes.

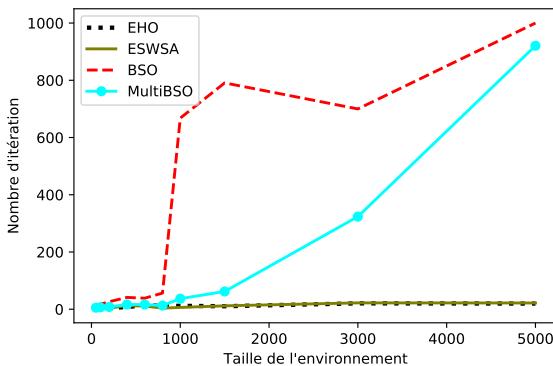


Figure 6.32: Comparaison de la variation du nombre d'itérations des algorithmes selon la taille de l'environnement (avec obstacles).

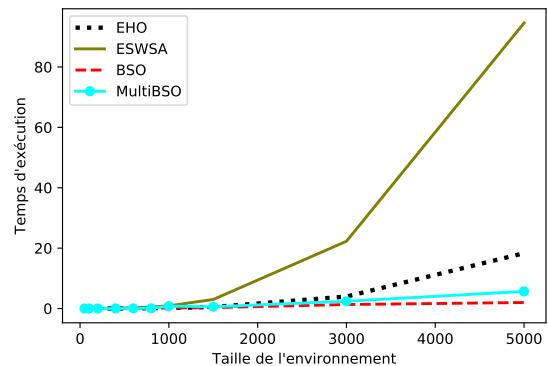


Figure 6.33: Comparaison de la variation du temps d'exécution des algorithmes selon la taille de l'environnement (avec obstacles).

c- Environnements complexes

- Mono-cible

Dans les figures 6.34 et 6.35 sont décrites la variation du nombre d'itérations et celle du temps d'exécution de nos algorithmes selon la taille des environnements complexes.

Nous pouvons regrouper les méta-heuristiques en deux groupes d'après l'évolution de leur nombre d'itérations, le 1^{er} groupe est constitué de BSO et Multi-BSO qui subit une importante croissance de ce nombre passant de 2.7 à 747.78 itérations pour BSO et de 2.05 à 444.45 itérations pour Multi-BSO.

Le 2^{ème} groupe est celui d'EHO et EWSA, ceux-ci croient de manière bien plus raisonnable, soient de 1.35 à 13.83 itérations pour EHO et de 1.05 à 70.45 itérations pour EWSA.

Toutes nos approches connaissent un accroissement des temps d'exécution avec l'élargissement des tailles d'environnement. EWSA est celle dont les temps ont le plus augmenté (de 0.001 à 12.32 secondes), suivie d'EHO avec des temps qui passent de 0.001 à 3.03 secondes, puis vient Multi-BSO avec entre 0.001 et 1.92 secondes. Enfin, la plus rapide est BSO, dont les temps sont entre 0.001 et 1 seconde.

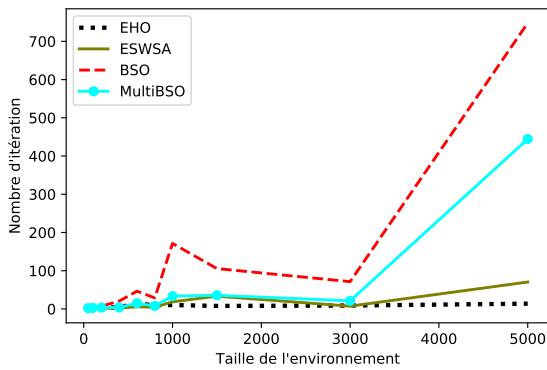


Figure 6.34: Comparaison de la variation du nombre d'itérations des algorithmes selon la taille de l'environnement (complexe).

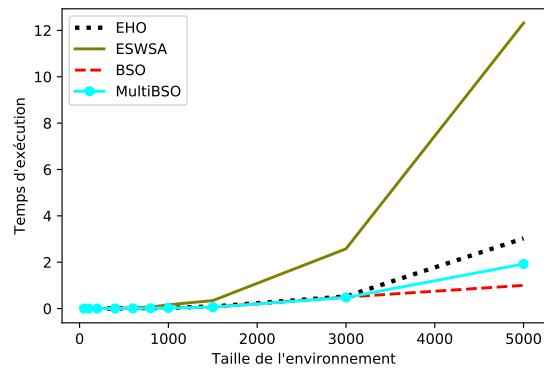


Figure 6.35: Comparaison de la variation du temps d'exécution des algorithmes selon la taille de l'environnement (complexe).

- Multi-cibles

Les courbes des figures 6.36 et 6.37 illustrent la variation du nombre moyen d'itérations et temps moyen d'exécution de nos algorithmes vis-à-vis des différentes tailles d'environnement complexes.

Du point de vue du nombre d'itérations, ils commencent tous avec environ 3 à 12 itérations pour le plus petit environnement puis augmentent. Ici aussi, c'est BSO et Multi-BSO qui enregistrent les plus grands nombres avec un maximum de 1000 et 865.5 itérations respectivement, ensuite EHO va jusqu'à 29.3 itérations et enfin EWSA ne dépasse pas la moyenne de 13.65 itérations.

Quant aux temps d'exécution les 4 approches évoluent de la même manière que pour le mode mono-cible, mais avec des temps plus importants, allant de 0.001 à 93.31 secondes pour EWSA, grimpant de 0.001 à 15.55 secondes pour EHO, puis de 0.001 à 10.23 secondes pour Multi-BSO, enfin pour BSO les temps vont de 0.001 à 4.79 secondes.

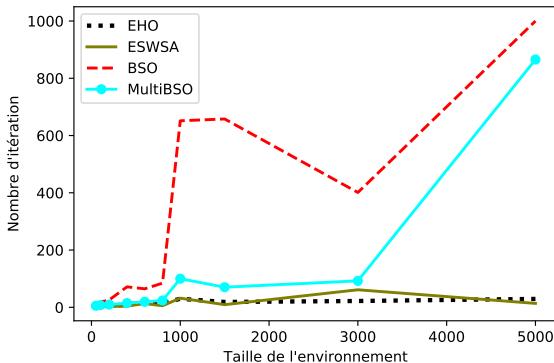


Figure 6.36: Comparaison de la variation du nombre d’itérations des algorithmes selon la taille de l’environnement (complexe).

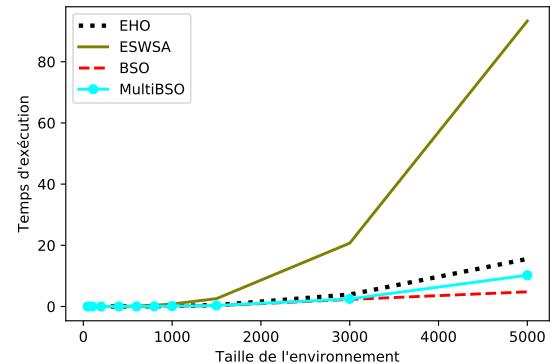


Figure 6.37: Comparaison de la variation du temps d’exécution des algorithmes selon la taille de l’environnement (complexe).

6.4.3.3 - Analyse relative à la taille des environnements

Suite aux résultats concernant la taille des environnements simples, avec obstacles et complexes selon les deux modes mono et multi-cibles, nous pouvons noter que :

- Contrairement aux deux approches inspirées des abeilles (BSO et Multi-BSO), les deux autres méta-heuristiques à savoir : EHO et ESWSA atteignent le taux maximum de réussite dans leur recherche de la ou les cible(s) quelle que soit la taille de l’environnement à explorer.
- Pour tous les algorithmes le nombre d’itérations croît avec l’augmentation de la taille des environnements, mais à des vitesses différentes. Telles qu’EHO et ESWSA effectuent le moins d’itérations.
- Les temps restent relativement acceptables pour le mono-cible, par contre pour le mode multi-cible ils atteignent les 1min 30. C’est ESWSA le plus long.
- EHO et ESWSA minimisent le nombre d’itérations en raison des grands pas entre deux positions successives qu’ils génèrent, mais cela leur coûte cher en temps, ce qui est l’inverse de BSO et Multi-BSO qui effectuent de petits pas dans des temps moindres.
- Pour les deux modes et pour toutes les approches, les temps d’exécution et nombre d’itérations croissent avec l’augmentation de la complexité des environnements.

6.4.4 Expérimentations par rapport au nombre de cibles

6.4.4.1 Taux de réussite

Les taux de réussite sont présentés sous forme de *diagramme à bandes* dans la figure 6.38, elle comporte les résultats de chaque méta-heuristique pour plusieurs nombres de cibles.

Nous nous apercevons qu'à l'unanimité toutes nos approches trouvent la totalité (100%) du nombre de cibles, quel que soit ce dernier entre 1 et 15 cibles (avec un pas de 2). Tout en respectant la limite du nombre d'itérations maximal (1000). Ces mêmes résultats sont valables pour les trois types d'environnement étudiés.

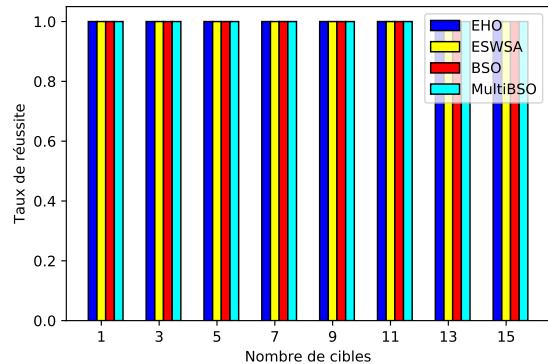


Figure 6.38: Comparaison de la variation du taux de réussite des algorithmes selon le nombre de cibles.

6.4.4.2 Variation du nombre d'itérations et temps d'exécution

a- Environnements simples (sans obstacles)

Les deux figures 6.39 et 6.40 ci-dessous retracent la variation du nombre moyen d'itérations et temps moyen d'exécution de nos algorithmes confrontés à un nombre croissant de cibles dans des environnements sans obstacles.

Tous les algorithmes résolvent le problème de recherche de cibles en un nombre d'itérations variant entre 1 à 31 itérations, à l'exception de BSO, qui vient bien après ses concurrents démarrant de 8 et arrivant jusqu'à 83 itérations. Notons que la meilleure performance est attribuée à EWSA.

Nous avons obtenu des temps d'exécution remarquablement réduits pour l'ensemble des méthodes, certes les temps ont accru avec l'augmentation du nombre de cibles, mais cela reste très raisonnable entre 0.02 et 0.05 secondes pour EWSA, entre 0.001 et 0.06 secondes pour Multi-BSO, de 0.01 à 0.07 secondes pour BSO et enfin, de 0.01 à 0.12 secondes pour ce qui est d'EHO.

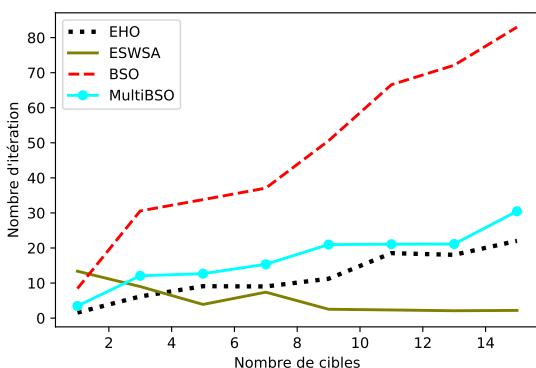


Figure 6.39: Comparaison de la variation du nombre d'itérations des algorithmes selon le nombre de cible (sans obstacles).

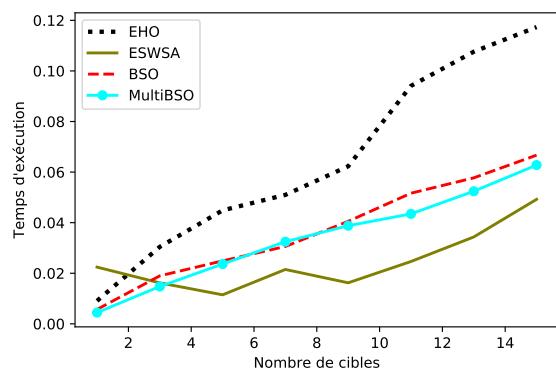


Figure 6.40: Comparaison de la variation du temps d'exécution des algorithmes selon le nombre de cible (sans obstacles).

b- Environnements avec obstacles

Les deux figures 6.41 et 6.42, décrivent la variation du nombre d’itérations et temps d’exécution de nos quatre approches à la recherche d’un nombre variable de cibles dans des environnements avec obstacles.

Le nombre d’itérations croît avec l’augmentation du nombre de cibles dans l’environnement, BSO connaît la plus grande croissance en passant de 23.53 à 104.43 itérations. Multi-BSO fait moins d’itérations avec entre 6.10 et 26.08. EHO et EWSA effectuent de 1.55 à 17.98 et de 2.58 à 7.58 itérations respectivement.

Les temps d’exécution sont très courts pour l’ensemble des méthodes, notons l’augmentation des temps de 0.02 jusqu’à 0.48 secondes pour EWSA. Ce dernier étant considéré comme le plus long, puis de 0.01 à 0.07 secondes pour EHO et de 0.01 à 0.05 pour les algorithmes inspirés des abeilles (BSO et Multi-BSO).

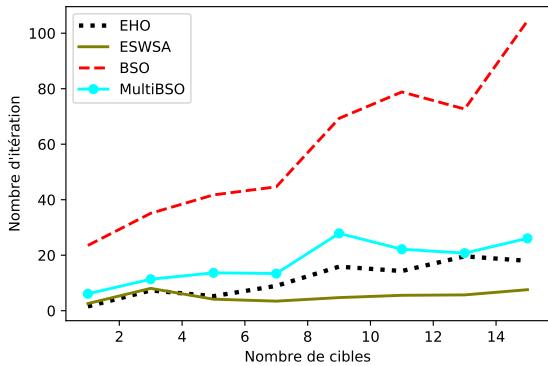


Figure 6.41: Comparaison de la variation du nombre d’itérations des algorithmes selon le nombre de cible (avec obstacles).

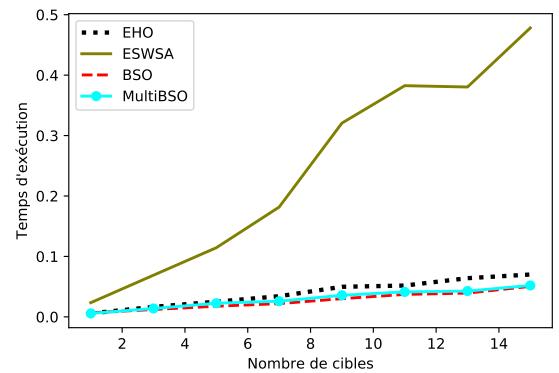


Figure 6.42: Comparaison de la variation du temps d’exécution des algorithmes selon le nombre de cible (avec obstacles).

c- Environnements complexes

Les *diagrammes à lignes brisées* des figures 6.43 et 6.44 mettent en relief l’influence du nombre de cibles recherchées sur le nombre d’itérations et temps d’exécution de nos algorithmes, dans des environnements complexes.

En observant le nombre d’itérations, nous constatons que nos approches peuvent être classées en fonction de leur rythme de croissance face au nombre de cibles recherchées. Telles que, BSO possède les plus grands nombres avec de 25.53 à 125.83 itérations, puis vient Multi-BSO qui passe de 8.73 à 35.28 itérations, suivies d’EHO dont les nombres sont entre 1.8 et 21.3 itérations, non loin EWSA avec entre 2.45 et 20.78 itérations.

Les algorithmes inspirés des abeilles possèdent des temps d’exécution bas, légèrement croissant avec l’augmentation du nombre de cibles passant de 0.01 à 0.05 secondes. EHO s’en rapproche avec une croissance de ses temps allant de 0.001 à 0.08 secondes. EWSA quant à lui sort du lot avec une augmentation de 0.02 à 0.48 secondes.

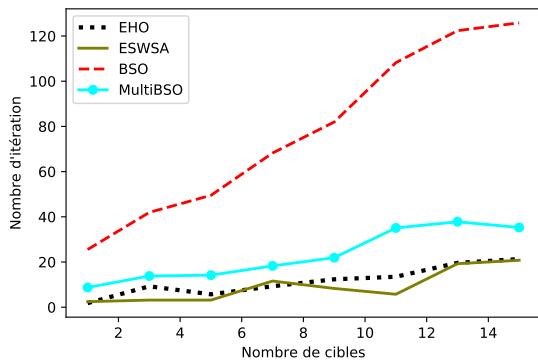


Figure 6.43: Comparaison de la variation du nombre d’itérations des algorithmes selon le nombre de cible (complexe).

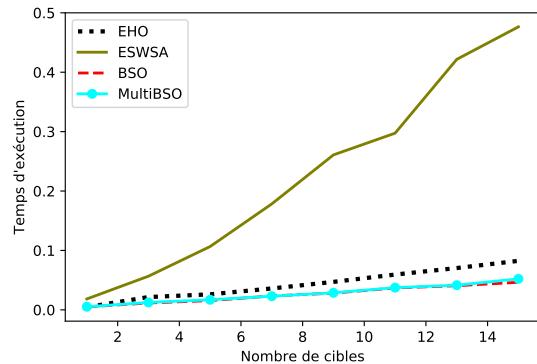


Figure 6.44: Comparaison de la variation du temps d’exécution des algorithmes selon le nombre de cible (complexe).

6.4.4.3 Analyse relative au nombre de cibles

D’après les résultats présentés ci-dessus par rapport au nombre de cibles recherchées dans les différents types d’environnements expérimentés, nous sommes en mesure de tirer les quelques conclusions qui suivent :

- Nos quatre mét-heuristiques ont fait leurs preuves de par leur capacité à trouver toutes les cibles qu’importe leur nombre (selon les conditions citées).
- Plus il y a de cible à chercher, plus les algorithmes font d’itérations et plus le temps d’exécution croît.
- BSO détient le plus grand nombre d’itérations quel que soit l’environnement, mais en contrepartie il possède les meilleurs temps d’exécution.
- La mét-heuristique EWSWA effectue des temps acceptables dans des environnements simples à grand nombre de cibles avec un nombre d’itérations record. En revanche, lors de la présence d’obstacles les temps d’exécution deviennent beaucoup trop importants comparés aux autres approches (BSO, EHO, Multi-BSO) détenant ainsi les pires temps.
- Quant aux deux autres algorithmes leur évolution en termes de temps et nombre d’itérations est bornée par celles de BSO et EWSWA.
- Le nombre d’itérations reste relativement acceptable pour 15 cibles, à noter qu’EWSWA, EHO et Multi-BSO étaient plus performants sur ce plan.
- Les temps d’exécution étaient globalement satisfaisants pour tous les algorithmes, car n’excédant pas les 0.5 secondes.
- La densité des environnements en obstacles, joue un rôle important dans le comportement de nos approches, telle que, plus nos environnements sont complexes plus nos approches produisent d’efforts et prennent de temps à atteindre les cibles.

6.4.5 Comparaison des types d'environnement

La moyenne des temps d'exécution et nombre d'itérations croît lors du passage d'environnements simples (sans obstacles) aux environnements avec obstacles, mais elles connaissent une considérable augmentation lorsqu'on a affaire à des environnements complexes.

Cela est dû à la complexité des environnements entravant et rendant plus difficile le mouvement des robots, ainsi le choix de la bonne trajectoire devient de plus en plus complexe ce qui impacte le temps d'exécution.

Pour ce qui est des taux de réussite, ils dépendent beaucoup plus des méthodes de recherche.

6.4.6 Comparaison de nos quatre approches

Nos quatre approches développées et testées ne possèdent pas les mêmes comportements face aux différentes variantes liées à l'environnement de recherche. Nous pouvons conclure que :

- L'approche Multi-BSO est la plus stable, les variations du nombre d'itérations et de temps d'exécution sont harmonieux.
- L'algorithme EWSA est le meilleur en termes de nombre d'itérations suivi de près par EHO.
- Les méta-heuristiques inspirées des abeilles sont les meilleures en termes de temps d'exécution.
- L'algorithme EWSA atteint parfois des temps d'exécution peu raisonnables.
- Globalement le multi-swarming (Multi-BSO) a grandement amélioré BSO que ce soit dans les taux de réussite, le nombre d'itérations ou temps d'exécution.
- BSO et Multi-BSO ont quelques lacunes par rapport aux très grands environnements.
- EHO possède le meilleur compromis entre taux de réussite (toujours à 100%), nombre d'itérations assez bas et temps d'exécution très raisonnables.

6.5 Simulateur

La réalisation de notre simulateur temps réel et interactif pour la visualisation de l'évolution des approches implémentées que ça soit BSO, Multi-BSO, EHO ou encore EWSA, a été mise en œuvre afin de faciliter la compréhension de notre travail.

6.5.1 Fonctionnement

L'affichage de l'environnement de recherche de notre simulateur passe par les étapes suivantes :

- Sélection de l'environnement à explorer sous sa forme matricielle comme décrite dans la modélisation de la section 3.2.
- Translation de l'espace de recherche sélectionné en une image PNG, en assignant aux obstacles et à la portée deux couleurs distinctes.
- Représentation de chaque position de l'environnement de recherche par un unique pixel dans l'image PNG.

Quant à l'étape de recherche des cibles, elle nécessite de retracer la trajectoire de chaque robot, en dessinant leurs chemins entre la position actuelle et la nouvelle position calculée par la méta-heuristique de recherche, le dessin suit l'algorithme de BRESENHAM [line].

Dans notre simulateur temps réel, pour une question de fluidité et de parallélisme, il est nécessaire de faire appel au *multi-threading*, un *thread* pour l'exécution de la méta-heuristique et un autre pour l'interface de simulation, la coordination entre ces deux *threads* est illustrée par le schéma de la figure 6.45 qui suit.

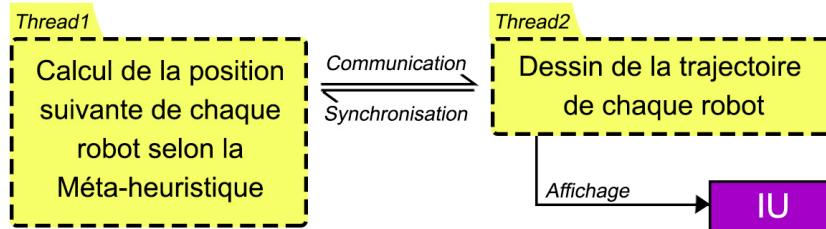


Figure 6.45: Schéma de communication du système multi-threads.

6.5.2 Interface

L'interface offre deux sections possibles, une dédiée au choix de l'environnement nous permettant de choisir un environnement selon nos préférences et paramètres, celle-ci est représentée dans la figure 6.46 ci-dessous :

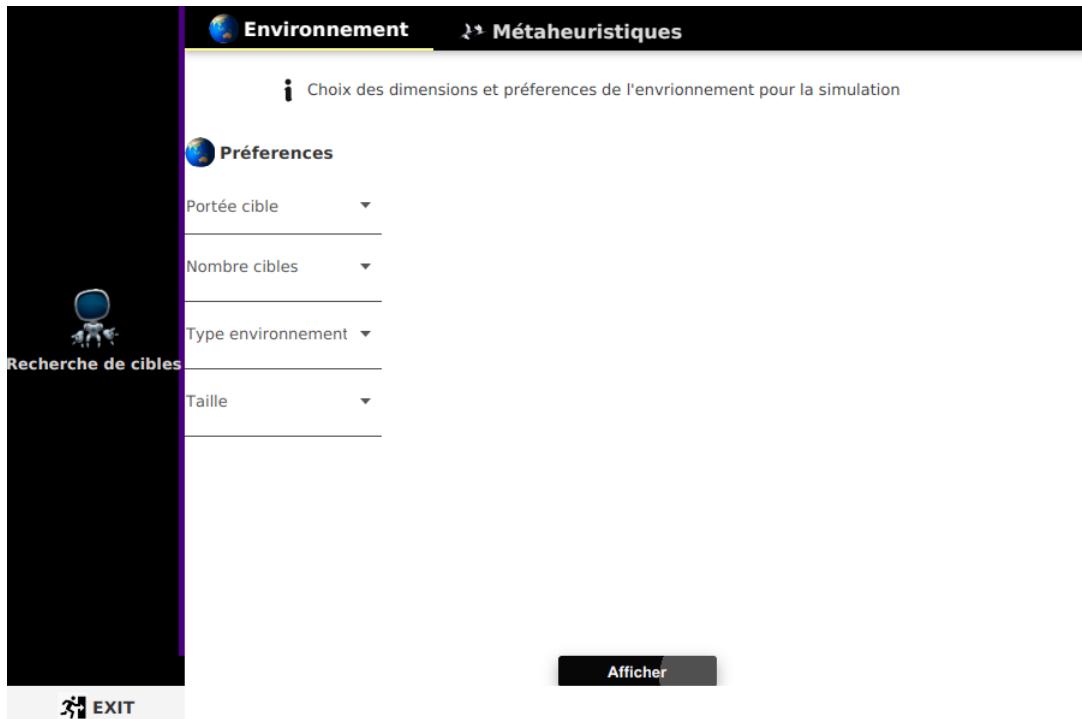


Figure 6.46: Présentation de l'interface

Et une autre section dédiée aux méta-heuristiques, qui englobe toutes les approches basées essaims traitées dans ce projet, afin d'être appliquées à l'environnement choisi dans la section Environnement. Cette section est visible dans la figure 6.47 qui suit :

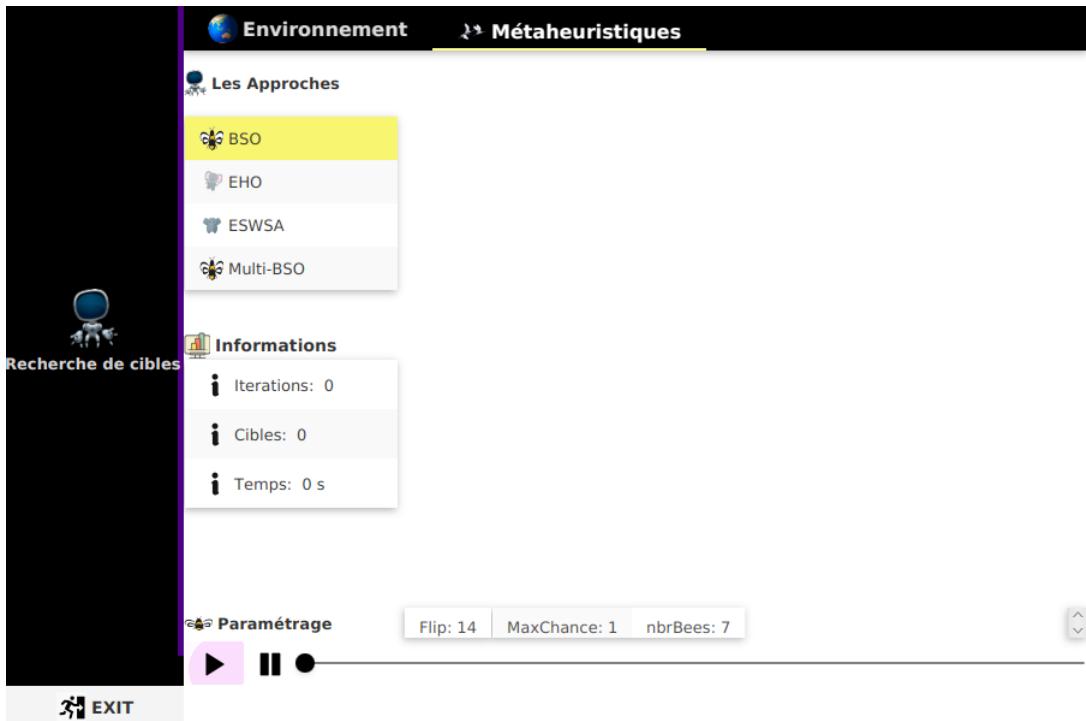


Figure 6.47: Présentation de la section Métaheuristiques

6.5.2.1 Environnement

La section Environnement permet à l'utilisateur de choisir un environnement selon les paramètres et préférences suivantes :

Nombre de cibles Nos environnements peuvent comporter une ou plusieurs cibles dont les positions sont générées de manière aléatoire.

Dans le cadre de notre simulation, nous nous sommes contentés de permettre deux choix pour le nombre de cibles qui sont:

- Mono-cible : une seule cible.
- Multi-cibles : avec 5 cibles.

Ce choix peut être augmenté selon le nombre de cibles souhaité.

Portée des cibles Les portées de cibles disponibles dans l'application sont donc catégorisées comme suit :

- **Portée petite**, d'un rayon d'environ 5% de la taille de l'environnement.
- **Portée moyenne**, d'un rayon d'environ 15% de la taille de l'environnement.
- **Portée large**, d'un rayon d'environ 25% de la taille de l'environnement.

Types d'environnements Les types d'environnements simples, avec obstacles et complexes sont représentés dans la figure 6.48, sont mis à disposition. Dans le cas des deux derniers types d'environnements, il est à noter que les positions et distributions des obstacles sont générées aléatoirement.

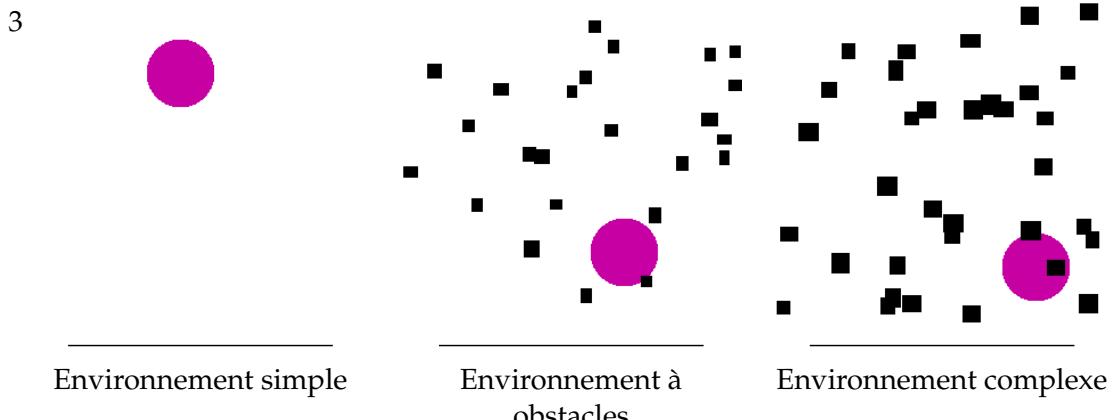


Figure 6.48: Les types d'environnements possibles.

Taille d'environnement Les environnements varient selon leur taille, on a alors choisi à titre représentatif quelques tailles par classe, telles que :

- 100 et 400 qui sont assez petits.
- 600 et 1000 dits de taille moyenne.
- 1500 qui est assez grand.

Afficher La section environnement présentée dans la figure 6.49 comporte le bouton *afficher* qui sert à visualiser l'environnement choisi en fonction des préférences insérées dans les listes de choix multiples de chaque influent (portée, nombre cibles, ...).

Ci-dessous un exemple d'environnement sélectionné :

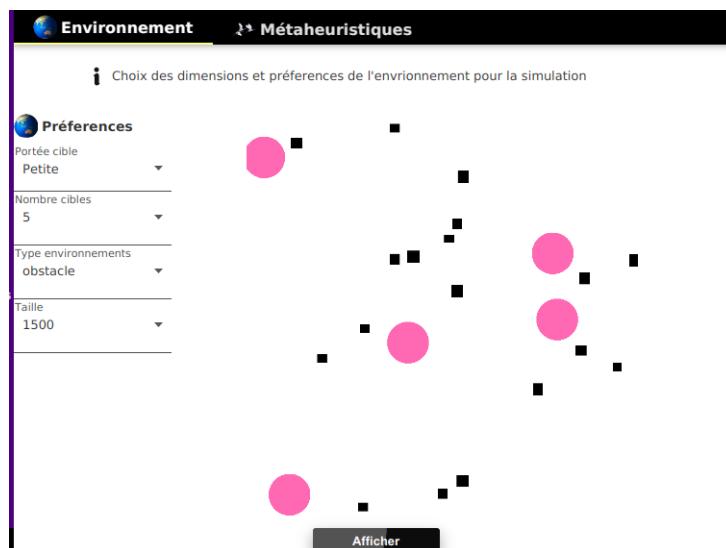


Figure 6.49: Affichage de l'environnement selon les préférences insérées

6.5.2.2 Métaheuristiques

Les approches Les approches basées essaims auxquels nous avons eu affaire tout au long de ce mémoire *BSO*, *EHO*, *ESWSA* et *Multi-BSO* y sont proposées sous forme d'une liste permettant de choisir celle à exécuter. Une fois qu'on clique sur une approche deux possibilités pour le choix du paramétrage s'offrent à nous:

- Paramétrage avec algorithme génétique L'option de paramétrage avec l'algorithme génétique (Mini-GA incrémental) est accessible pour chaque approche, comme le montre la figure 6.50 ci-dessous.

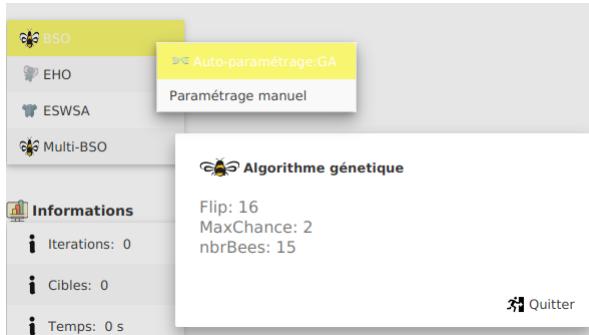


Figure 6.50: Auto-paramétrage avec l'algorithme génétique

- Paramétrage manuel Le paramétrage manuel se fait en choisissant pour chaque paramètre de l'approche basée essaim une valeur. Les figures 6.51 et 6.52 ci-dessous illustrent ces paramètres pour BSO.

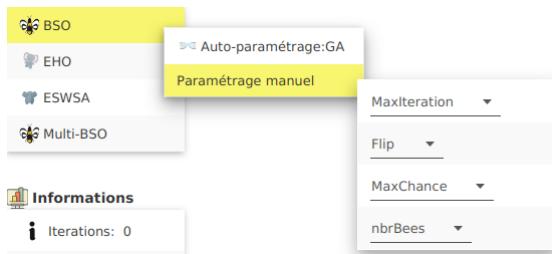


Figure 6.51: Menu de paramétrage

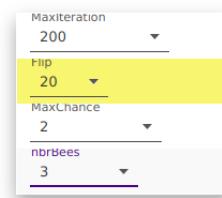


Figure 6.52: Menu de paramétrage manuel pour BSO

Les informations Ce sont les détails identiques pour chaque approche et qui sont : *Nombre d'itérations*, *Nombre de cibles trouvées*, *Temps d'exécution (sec)* comme présentés dans la figure 6.53. Ces informations sont mises à jour en temps réel, c'est-à-dire qu'elles sont modifiées à chaque itération (le nombre d'itérations inclus).



Figure 6.53: Informations durant l'exécution de BSO

Paramétrage est une liste horizontale comme illustrée dans la figure 6.54, elle contient les paramètres de l'exécution de l'approche choisie, qu'elle soit générée par algorithme génétique ou bien manuellement.



Figure 6.54: Liste des paramètres pour BSO

Start Le bouton "Start" ➤ permet de débuter l'exécution après avoir choisi l'approche. Ainsi, l'image de l'environnement et des robots (abeilles ou éléphants) sur l'environnement s'affichent.

Stop Le bouton "Stop"  permet d'arrêter et mettre fin à l'exécution de l'approche, afin de choisir une autre méta-heuristique à exécuter ou bien relancer la même avec d'autres paramètres.

Le slider de la figure 6.55 joue le rôle d'une barre de progression, tel qu'il s'incrémenter après chaque itération de l'approche en cours d'exécution. Une fois l'exécution terminée, il permet de revenir en arrière retracant ainsi la trajectoire des robots dans l'environnement. Comme pour un film ou une vidéo, nous avons la possibilité d'avancer du début jusqu'à la fin, pour revoir l'exécution.



Figure 6.55: Le slider

L'illustration 6.56 suivante est un exemple d'une exécution de BSO dans un environnement multi-cibles de petites portées et avec obstacles.

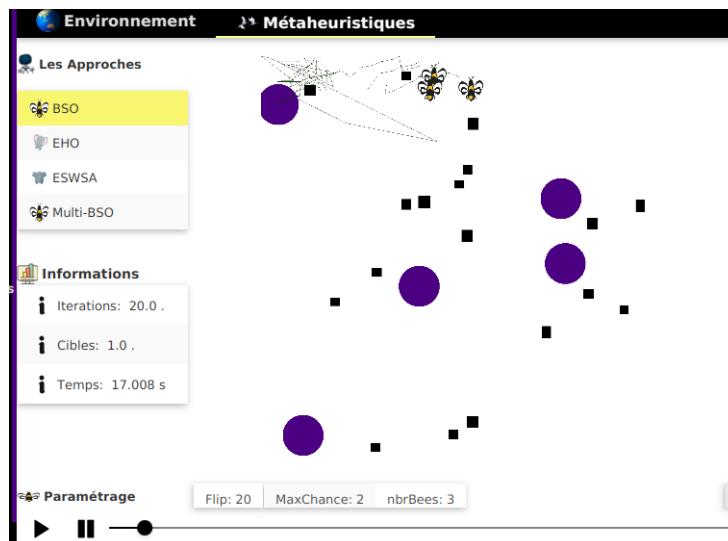


Figure 6.56: Exemple d'exécution de BSO.

Quitter Un bouton "Quitter"  se trouve en bas à gauche de l'interface, il permet de sortir de l'application dès que nous le souhaitons.

6.6 Conclusion

À travers ce chapitre, on a pu discuter et effectuer une multitude de tests expérimentaux qui concernent toutes les approches de résolution d'intelligence en essaim implémentées qu'on cite BSO, Multi-BSO, EHO et EWSA. D'autres tests ont été présentés portant sur l'algorithme génétique qui ont comme objectif d'effectuer un réglage de paramètres des algorithmes implémentés.

La série de tests relative à l'environnement (23 040 exécutions) a touché à plusieurs caractéristiques comme *La portée de la cible, la taille de l'environnement, nombre de cibles ainsi que la densité des obstacles*. L'analyse de l'ensemble des résultats nous a permis d'observer les changements de comportement des algorithmes, ce qui nous a aidées à extraire les avantages de chaque approche, ainsi que leurs inconvénients tout en justifiant les causes.

Ainsi, nous envisageons une hybride qui peut éventuellement surpasser les algorithmes vus dans ce mémoire, cela en combinant leurs points forts et remédiant à leurs points faibles.

Conclusion générale et perspectives

La recherche de cibles reste un problème attractif, sujet à de multiples propositions de résolution qui ont souvent recours aux systèmes multi-robots. Ces derniers se trouvant dans des environnements complexes et inconnus, doivent être dotés d'une stratégie d'évitement d'obstacles pour évoluer dans un monde inconnu dans lequel ils doivent chercher des cibles.

Le développement de méthodes de recherche efficaces, optimisées et rapides est une nécessité dans laquelle s'inscrit notre travail.

Pour ce faire, il nous a fallu en premier lieu nous familiariser avec le domaine de la recherche de cibles et avec ses nombreuses variantes et paramètres. Cette étape nous a permis de définir dans quelle branche notre présent projet prend place. Par la suite nous avons présenté une synthèse de quelques travaux relatifs à notre problématique, pour mieux cerner le problème et nous informer des méthodes les plus adaptées à sa résolution.

En second lieu nous nous sommes penchées sur les outils de résolution basés sur l'intelligence en essaim, à savoir les méta-heuristiques s'inspirant des abeilles (BSO et Multi-BSO) et des éléphants (EHO et EWSA) ou encore de la génétique (GA). Ce choix fut motivé par les caractéristiques de ces différentes méthodes qui s'adaptent au mieux à la formulation du problème auquel nous sommes confrontées.

Par ailleurs, nous avons clairement défini notre modélisation par rapport à l'environnement, aux cibles et aux robots. Puis il nous a fallu choisir une stratégie d'évitement d'obstacles conforme à cette modélisation. Ce choix s'est porté sur la méthode d'échantillonnage, pour enfin décrire formellement le fonctionnement des quatre approches de recherche qui constituent nos deux contributions majeures, ainsi que l'algorithme génétique mis à profit pour le réglage automatique des paramètres.

À l'issue de cette phase, nous sommes passées à l'implémentation de nos algorithmes sur machine.

Pour visualiser et suivre la recherche, étape par étape, nous avons mis en place un simulateur temps réel, exploitant le multi-threading pour permettre une application fluide.

Enfin, l'évaluation de nos approches a été appuyée par des expérimentations multiples et répétées, réalisées de manière comparative, qui ont montré leur efficacité et leur fiabilité. L'analyse de nos résultats, nous a permis de mettre en évidence les avantages et les inconvénients de chaque approche. Notre ambition est d'exploiter ces différentes approches à travers une hybridation des quatre algorithmes basés essaim, ne retenant que les atouts de chacun, pour optimiser au mieux les résultats de notre recherche.

Pour résumer, dans le présent travail nous avons développé les modules suivants:

- Un générateur d'environnement de recherche.
- L'approche BSO pour la recherche de cibles.
- L'approche Multi-BSO pour la recherche de cibles.

- L'approche EHO pour la recherche de cibles.
- L'approche EWSA pour la recherche de cibles.
- L'algorithme GA de paramétrage.
- La stratégie d'évitement d'obstacles et de simulation du déplacement des robots.
- L'auto-génération des expérimentations en graphiques.
- Un simulateur de recherche temps réel.

Néanmoins, nous pensons déjà à quelques améliorations que nous avons en perspective, dont nous citerons :

L'adaptation de la recherche aux cibles mobiles : Les cibles mobiles requièrent une première étape de détection comme vue dans le présent travail, mais aussi un suivi de ces cibles, car leurs positions changent dans le temps.

La prise en charge des obstacles mobiles : L'adaptation de notre stratégie d'évitement d'obstacles par échantillonnage aux obstacles mobiles, tels que: les êtres humains, animaux, voitures, ...), car cela nous permet de nous rapprocher de la complexité réelle du monde dans lequel nous vivons.

Passer à la version multi-agents de nos approches : Afin de bénéficier du parallélisme dont devrait disposer un système multi-robots ainsi que d'une coopération plus réaliste.

Hybridation des approches : Les résultats obtenus nous ont permis d'extraire les points forts de chaque approche, par exemple:

EHO : L'organisation des robots en plusieurs clans, mais aussi l'opérateur de séparation qui permet la diversification lors de la recherche.

BSO : Le flip, permettant un espacement homogène entre les robots, ce qui réduit la recherche redondante des robots pour une même zone. Ainsi que le critère de diversité comme solution à la stagnation.

EWSA : W^t pour la gestion de la vitesse des déplacements. Le déplacement de manière continue grâce à la vitesse.

Multi-BSO : Liste Tabou commune à tous les groupes de robots, exploitée comme un tableau noir.

Partitionnement de l'environnement, en affectant des équipes de robots à chaque partition, ce qui permettra de réduire à zéro le repassage des groupes de robots par les mêmes partitions.