REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
ET DE LA RECHERCHE SCIENTIQUE

UNIVERSITE LARBI BEN M'HIDI – OUM EL BOUAGHI
FACULTE DES SCIENCES EXACTES ET DES SCIENCES DE LA NATURE
ET DE LA VIE
Département Mathématique et Informatique

N° D'ordre :……

Série :……..

Mémoire pour obtenir le diplôme de magister en Informatique

Option : Intelligence Artificielle et Imagerie

OPTIMISATION DYNAMIQUE PAR DES APPROCHES INSPIREES DE LA NATURE

Présenté par : DAAS Mohamed Skander

Soutenu le : …./…./2014

Devant le jury composé de :

Président : Pr. BENMOHAMMED Mohamed, Prof à l'université Constantine2

Rapporteur : Pr. BATOUCHE Mohamed, Prof à l'université Constantine2

Examinateurs : Dr. BOUTEKKOUK Fateh, MC.A à l'université Oum El Bouaghi

Dr. NINI Brahim, MC.A à l'université Oum El Bouaghi

2013/2014

Remerciements

Avant tout, j'adresse mes sincères remerciements à mon encadreur ***Pr Mohamed Chaouki BATOUCHE*** *pour son soutien tout au long de ces deux années de thèse. Je ne saurais dire combien son aide et ses conseils m'ont été précieux.*

Je tiens à exprimer toute ma gratitude au ***Dr Farid Mokhati*** *Responsable de notre formation de post-graduation (Magister) qui nous a beaucoup aidé.*

Je tiens à remercier ***Pr Mohamed Benmohammed*** *professeur à l'université de Constantine 2 pour l'honneur qu'il me fait en acceptant d'être président de jury.*

Je suis très reconnaissant aux ***Dr Fateh Boutekkouk*** *et* ***Dr Brahim NINI*** *d'avoir participés à mon jury de thèse et les en remercie sincèrement.*

J'adresse aussi mes vifs remerciements à tous mes enseignants qui ont contribués à ma formation.

Je remercie mes collègues et amis de Constantine et d'Oum El Bouaghi pour leur sympathie et leur bonne humeur ainsi que toute l'équipe de la DJS.

Je voudrais remercier aussi et très spécialement ***H. ISMAHENE*** *pour son appui et ses encouragements.*

Bien sûr, je ne peux terminer sans remercier mes proches de tout cœur et notamment mes ***parents*** *qui, au cours de ces deux années de thèse, m'ont toujours soutenu et encouragé, comme d'habitude...*

الملخص

تتناول هذه المذكرة مشكلة التحسين الديناميكي باستخدام نهج مستوحاة من الطبيعة. أولا، نقترح اثنين من الخوارزميات DBFO و DABFOعلى أساس BFO مستوحاة من السلوك الذكي للبكتيريا، وقد استخدمت عدة أساليب لتكييف الخوارزم الأساسي للبيئات الديناميكية، ومن ثم يتم تطبيق هذه الخوارزميات لحل مشكلة تتبع الأجسام باستخدام تشابه الرسوم البيانية. باستخدام التقنيات المطبقة، DABFO يفوق الخوارزميات القائمة على أساس BFO خصوصا عندما تكون شدة وسرعة التغيير كبيرة. ثانيا، نقترح نهجا متعدد الشعوب(MBFO) ، استنادا إلى BFO الذي يستخدم آليات الإقصاء والتنوع تتكيف مع البيئات الديناميكية. على الرغم من أن النتائج التجريبية التي تم الحصول عليها من خلال الدراسات المقارنة لا تظهر تفوق الخوارزمية MBFO المقترحة في الحالة العامة، ولكن MBFO يظهر أداء أفضل في البيئات التي تتميز بترداد تغيير عالية.

الكلمات الرئيسية: التحسين الديناميكي، النهج المستوحاة من الطبيعة، BFO، MBFO، DABFO، تعدد الأسراب ، التنوع، تتبع الأجسام

Résumé

Ce mémoire traite le problème de l'optimisation dynamique en utilisant des approches inspirées de la nature. Dans un premier temps nous proposerons deux algorithmes DBFO et DABFO basés sur le comportement intelligent des bactéries. Plusieurs techniques ont été utilisées pour adapter l'algorithme de base aux environnements dynamiques. Ces deux versions d'algorithmes ont été ensuite appliquées pour traiter le problème de la poursuite d'objets en utilisant la similarité des histogrammes. Grâce aux techniques appliquées, DABFO rivalise et surpasse même les algorithmes existants basés BFO surtout quand la gravité et la vitesse des changements sont grandes. Dans un second temps nous proposons une approche multi-population (MBFO) basée-BFO, qui utilise des mécanismes d'exclusion et de diversité adaptés aux environnements dynamiques. Bien que les résultats expérimentaux obtenus à traves les études comparatives ne montrent pas dans le cas général la supériorité de notre algorithme proposé (MBFO), mais MBFO obtient de meilleurs résultats dans des environnements caractérisés par une large fréquence de changement.

Mots clés : optimisation dynamique, approches inspirées de la nature, BFO, DABFO, MBFO, multi-population, diversité, poursuite d'objets.

Abstract

This thesis addresses the problem of dynamic optimization using nature inspired approaches. First, we propose two algorithms DBFO and DABFO based on BFO (based on the intelligent behavior of bacteria). Several techniques have been used to adapt the basic algorithm for dynamic environments, and those two versions of algorithms are then applied to resolve the problem of object tracking using histograms' similarity. Using applied techniques, DABFO rivals and even surpasses the existing algorithms based on BFO especially when the severity and speed of change are large. Second, we propose a multi-population approach (MBFO) based on BFO which uses exclusion and diversity mechanisms adapted to dynamic environments. Although the experimental results obtained through comparative studies do not show the superiority of our proposed algorithm MBFO in the general case, but MBFO performs better in environments which are characterized by a high change frequency.

Keywords : dynamic optimization, nature inspired approaches, BFO, MBFO, DABFO, multi-swarm, diversity, objet tracking,.

Sommaire

Remerciements

Résumé **1**

Abstract **2**

Table des matières **3**

Liste des figures **6**

Liste des tableaux **7**

1 - Introduction

1.1 Problèmes de l'optimisation dynamique 8
1.2 Approches naturo-inspirées pour l'optimisation dynamique 9
1.3 Contribution 10
1.4 Organisation du mémoire 11

2 - Optimisation dynamique

2.1 Introduction 13
2.2 Fonction objectif 14
2.3 Théorie de la complexité 14
2.4 Définition formelle de l'optimisation 15
2.5 Classification des algorithmes d'optimisation 16
2.5.1 Algorithmes probabilistes et algorithmes déterministes 17
2.5.2 Algorithmes exactes et Algorithmes approchés 18
2.6 Quelques problèmes rencontrés dans l'optimisation globale 19
2.6.1 Convergence prématurée 19
2.6.2 Exploration et exploitation 19
2.6.3 Changement dynamique des problèmes 20
2.7 Optimisation dynamique 20
2.8 Classification des environnements dynamiques 21
2.9 Caractéristiques des environnements dynamiques 22
2.10 Définition formelle de l'optimisation dynamique 23
2.11Techniques utilisées par l'optimisation dynamique 24
2.11.1 Introduire la diversité à la détection du changement 24

2.11.2 Maintenir la diversité durant la recherche 24
2.11.3 Utiliser la mémoire 24
2.11.4 Utiliser la prédiction 25
2.11.5 Utiliser plusieurs populations 26
2.12 Quelques problèmes d'optimisation dynamique 26
2.12.1 Problème dynamique du voyageur de commerce D-TSP 26
2.12.2 Problème dynamique du sac à dos D-KNAPSACK 27
2.12.3 Problème dynamique de coloration de graphe D-GC 27
2.12.4 Problème dynamique de satisfiabilité maximale (D-MAX-SAT) 28
2.12.5 Problème du MPB (Moving Peaks Benchmark) 28
2.13 Conclusion 30

3 - Approches naturo-inspirées pour l'optimisation dynamique
3.1 Introduction 32
3.2 Quelques approches inspirées de la nature 32
3.2.1 Les approches basées sur les algorithmes évolutionnaires 33
3.2.1.1 Les algorithmes génétiques GA 34
3.2.1.2 Programmation génétique GP 34
3.2.1.3 Stratégies d'évolution ES 35
3.2.1.4 Programmation évolutionnaire EP 35
3.2.1.5 Évolution différentielle DE 35
3.2.1.6 Etat de l'art sur les approches évolutionnaires dynamiques 38
3.2.3 Les approches basées sur l'intelligence distribuée 44
3.2.3.1 Optimisation par essaims particulaires PSO 46
3.2.3.2 Algorithme de colonie de fourmis ACO 47
3.2.3.3 Optimisation par recherche de nourriture bactérienne BFO 49
3.2.3.4 Algorithme de lucioles FA 53
3.2.3.5 Systèmes immunitaires artificiels AIS 56
3.2.3.6 Etat de l'art sur les approches dynamiques de l'intelligence distribuée 57
3.3 Conclusion 59

4 - Nouvelles approches pour l'optimisation dynamique basées BFO
4.1 Introduction 60
4.2 BFO dynamique auto-adaptative 61
4.2.1 Elimination-dispersion Auto-adaptive 61
4.2.2 Prédiction auto-adaptative 62
4.2.2.1 La position de la région de prédiction 62
4.2.2.2 La taille de la région de prédiction 63
4.2.2.3 Le nombre d'individus destinés à la prédiction 63
4.2.3 Application du DBFO et DABFO pour le problème de la poursuite d'objets ... 65
4.2.1.1 Histogramme de couleur 65
4.2.1.2 Formulation de la fonction objectif 67
4.2.4 Expériences et résultats 67

4.2.4.1 Expériences et configurations de l'environnement 67
4.2.4.2 Métriques de performances utilisées 68
4.2.4.3 Résultats expérimentaux 68
4.2.5 Discussion et analyse des résultats 70
4.2.6 Conclusion 72
4.3 BFO multi-population proposé 73
4.3.1 Exclusion dans MBFO 73
4.3.2 Réactivité aux changements et maintien de la diversité 74
4.3.3 Pseudo-code du MBFO 75
4.3.4 Expériences et résultats 76
4.3.4.1 La fonction de test dynamique 76
4.3.4.2 Métriques de performances utilisées 77
4.3.5 Résultats expérimentaux 77
4.3.6 Discussion et analyse des résultats 79
4.3.7 Conclusion 80

5 - Conclusion et perspectives
5.1 Conclusion 81
5.2 Perspectives 82

Bibliographie 83
Annexe 1: Publication "**A Dynamic Auto-Adaptive Bacterial Foraging Optimization for Color-Based Object Tracking**". Proceeding of the 2nd International Conference on Segnal, Image, Vision and their Applications (SIVA'13), November 18-20, 2013 - Guelma, Algeria, pp. 413-418"**.**

Liste des figures

1.1 Quelques inspirations de la nature 9
2.1 Complexité des problèmes 15
2.2 Les optima globaux et locaux d'une fonction à deux dimensions 16
2.3 La taxonomie des algorithmes d'optimisation globale 18
2.4 La convergence prématurée dans un espace d'une fonction objectif 2D 19
2.5 Classification des problèmes de l'optimisation dynamique 21
2.6 Exemple de la technique de prédiction 25
2.7 Exemple de la technique multi-population 26
2.8 Problème du TSP 27
2.9 Problème du sac à dos 27
2.10 Problème de coloration de graphe 28
2.11 Problème du MPB 29
3.1 Exemple du croisement 33
3.2 Exemple de la mutation 33
3.3 Représentation abstraite de l'arbre syntaxique des algorithmes / programmes 34
3.4 Un exemple d'une fonction objectif à 2-dimensions qui montre ses contours et le processus de génération du vecteur mutant 36
3.5 Illustration du processus de croisement pour d = 7 paramètres 37
3.6 Intelligence distribuée 44
3.6 Description de la vitesse et des mises à jour des positions dans l'optimisation par PSO pour un espace de paramètres à deux dimensions 46
3.7 Recherche des fourmis du plus court chemin de la nourriture vers le nid en suivant la phéromone 48
3.8 La nage, les culbutes et le comportement chimiotactique de l'E. Coli 49
3.9 Exemple du processus de la Chimiotaxie 50
3.10 Attractivité des lucioles par la lumière 54
3.11 Les Lymphocytes (comme les cellules B et T) reconnaissent et éliminent les antigènes dans un système immunitaire naturel 56
4.1 Prédiction auto-adaptative 62
4.2 Exemple d'une fenêtre de la sous région candidate d'une image de la vidéo 66
4.3 Histogramme du niveau teinte 67
4.4 Exemple de trames de 3 vidéos du benchmark BoBoT 68
4.5 Fitness moyen collectif de la vidéo A 69
4.6 Fitness moyen collectif de la vidéo B 69
4.7 Fitness moyen collectif de la vidéo C 70
4.8 Quelques applications du DABFO 71
4.9 Exemple de l'exclusion dans le model multi-population 74

Liste des tableaux

4.1 Fitness moyen collectif..... 70
4.2 Statistiques sur le nombre de trames mieux poursuivies..... 70
4.3 Paramètres par défaut du MBP..... 76
4.4 L'erreur hors ligne et l'erreur standard pour f=500 78
4.5 L'erreur hors ligne et l'erreur standard pour f=1000 78
4.6 L'erreur hors ligne et l'erreur standard pour f=2500 79
4.7 L'erreur hors ligne et l'erreur standard pour f=5000 79

Chapitre 1

Introduction

Les problèmes de l'optimisation dynamique existent dans plusieurs domaines du monde réel. Nous pouvons les rencontrer par exemple dans le contrôle du trafic aérien, dans la poursuite de cibles, dans les applications militaires et dans bien d'autres domaines d'applications. Bien qu'il existe plusieurs approches qui résolvent avec succès les problèmes d'optimisation statique, les problèmes d'optimisation dynamique posent un réel défi pour ce genre d'algorithmes. L'objet de ce mémoire consiste donc à étudier d'une part les mécanismes inspirés de la nature utilisés pour traiter le problème de l'optimisation dynamique et d'autre part, il s'agit de proposer et de concevoir de nouvelles approches et les faire appliquer à des problèmes du monde réel.

1.1 Problème de l'optimisation dynamique

Plusieurs problèmes d'optimisation du monde réel incorporent des objectifs, des contraintes, et des paramètres qui changent constamment avec le temps. Le changement dans le problème peut être lié à la fonction objectif, aux contraintes, aux bornes des variables ou bien à toute combinaison de ces derniers, ce genre de problèmes devrait être résolu instantanément à chaque fois qu'il y aura un changement. Cependant, en pratique, une tâche d'optimisation nécessite un certain temps pour atteindre une solution qui est raisonnablement proche de la solution optimale mais en même temps le problème peut avoir changé.

Les problèmes d'optimisation dynamique peuvent être résolus de plusieurs manières de la même façon que les problèmes d'optimisation statique [Gregory 2011]. Pour certains problèmes, les algorithmes peuvent être en mesure de trouver des solutions optimales pour tous les environnements possibles, pour d'autres problèmes, un algorithme doit s'adapter rapidement aux changements de l'environnement en se basant sur très peu d'informations du problème.

1.2 Approches naturo-inspirées pour l'optimisation dynamique

Une grande classe d'algorithmes couramment utilisés pour les problèmes dynamiques sont des méta-heuristiques inspirées de la nature (les algorithmes évolutionnaires, l'optimisation par colonie de fourmis, l'optimisation par essaim particulaire …etc.)

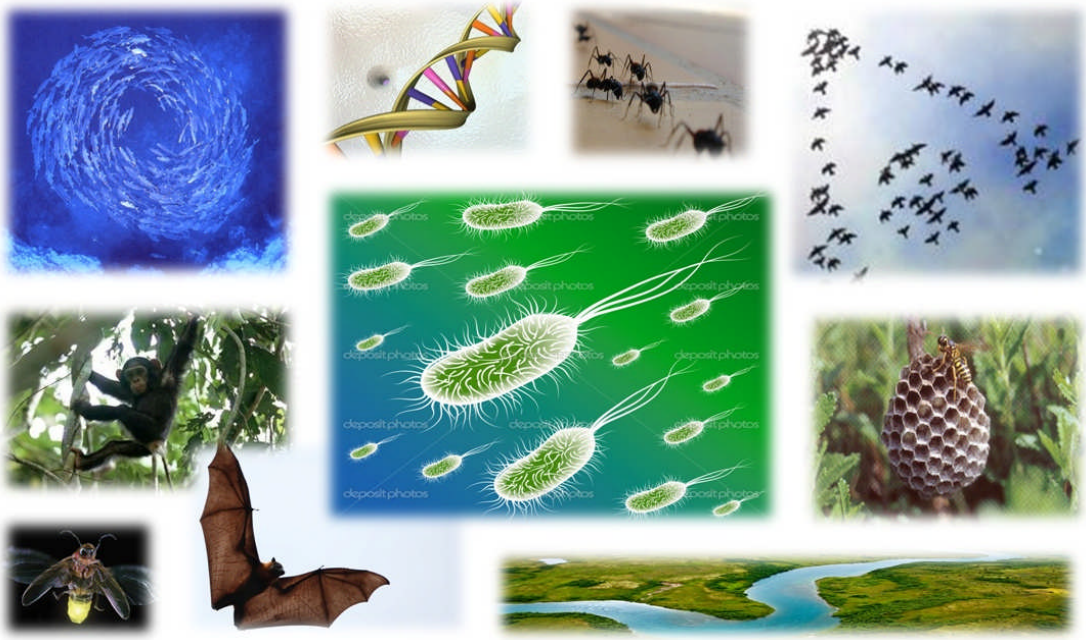

Figure 1.1: Quelques inspirations de la nature.

Contrairement aux méthodes classiques, les méta-heuristiques ne nécessitent généralement pas de connaissances approfondies de la fonction à optimiser comme les emplacements de bonnes solutions ou des dérivés de la fonction à optimiser. Ceci est particulièrement avantageux, car la plupart des problèmes dynamiques sont difficiles en particulier en raison du manque d'informations sur la fonction à optimiser.

Le but d'une méta-heuristique est de résoudre un problème d'optimisation donné; elle cherche un objet mathématique (une variable, un vecteur … etc) minimisant (ou maximisant) une fonction objectif qui décrit la qualité d'une solution pour un problème. L'ensemble de solutions possibles forme l'espace de recherche. L'espace de recherche est au minimum borné, mais peut être également limité par un ensemble de contraintes. Les méta-heuristiques manipulent une ou plusieurs solutions à la recherche de l'optimum. Les itérations successives doivent permettre de passer d'une solution de mauvaise qualité à une meilleure pour un

problème donné. L'algorithme s'arrête après avoir atteint un critère d'arrêt, consistant généralement à l'atteinte du temps d'exécution imparti ou à une certaine précision demandée.

Une solution ou un ensemble de solutions est parfois appelé un état que la méta-heuristique se fait évoluer via des transitions ou des mouvements. Si une nouvelle solution est construite à partir d'une solution existante, elle est sa voisine. Le choix du voisinage et de la structure des données peut être crucial. Dans certains cas, le but recherché est explicitement de trouver un ensemble d'optimums, l'algorithme doit alors trouver l'ensemble des solutions de bonnes qualités, sans nécessairement se limiter à un seul optimum : on parle de méthodes multimodales.

Malgré que les approches d'optimisation inspirées de la nature sont principalement conçues pour résoudre des problèmes d'optimisation statique plutôt que leurs homologues dynamiques [Tim 2009], plusieurs travaux ont été réalisés sur des versions améliorées de ces méta-heuristiques en ajoutant quelques techniques afin de leurs permettre de résoudre les problèmes d'optimisation dynamique.

1.3 Contribution

Dans ce travail, nous présentons deux nouvelles approches, La première approche que nous allons présenter est un algorithme dynamique basé BFO (Bacterial Foraging Optimization), cette version dynamique de cet algorithme est étendue à une deuxième version auto-adaptative nommée (DABFO) en introduisant des techniques adaptatives de prédiction et de maintien de la diversité. Ces techniques sont basées sur la trajectoire, la vitesse et l'évolution de la valeur du meilleur fitness déjà trouvée. L'intégration de ces techniques proposées assure une bonne diversité et un compromis exploration-exploitation plus efficace sur l'espace de recherche dans un environnement dynamique. Les deux versions d'algorithmes (DBFO et DABFO) sont appliquées pour traiter le problème de la poursuite d'objets en utilisant la similarité des histogrammes.

La deuxième approche est un algorithme multi-population basé BFO. L'idée principale est de diviser la population des individus en plusieurs sous populations. Chaque sous population suit les mêmes règles que l'algorithme de base BFO avec quelques modifications sur le mécanisme de déclanchement de l'étape d'élimination et de dispersion sur certaines conditions liées au dynamisme environnemental. Afin de garantir une bonne exploration de l'espace de recherche et de poursuivre plusieurs optima, autres mécanismes inter-populations

ont été aussi intégrés telle que le mécanisme de l'exclusion entre les sous populations qui a prouvé son efficacité dans plusieurs travaux. Pour améliorer la recherche locale et accélérer la convergence vers les régions les plus prometteuses, l'exploitation est renforcée par l'élimination de quelques populations dans le but de les faire réinitialiser. Notre algorithme BFO multi-population est évalué avec des scénarios du benchmark multimodal et dynamique MPB (Moving Peaks Benchmark). Les résultats des simulations sont comparés à ceux d'autres approches de la littérature. Les résultats montrent que l'algorithme proposé peut concurrencer certaines versions et même surpasser d'autres.

1.4 Organisation du mémoire

Après la présente introduction (chapitre 1) nous allons d'abord présenter l'optimisation globale dans le chapitre 2 qui fait appel à quelques notions sur l'optimisation et la théorie de la complexité, puis, nous nous focaliserons sur l'optimisation dans les environnements dynamiques et nous nous intéresserons aux différentes techniques utilisées par les différentes approches pour traiter ce problème. Les différentes versions dynamiques de quelques problèmes classiques sont aussi présentées dans ce chapitre ainsi qu'une présentation détaillée du problème MPB (Moving Peaks Bebchmark) qui sera utilisé comme un benchmark dans la partie expérimentale dans le chapitre 4.

Dans le chapitre 3, nous présentons les différentes approches inspirées de la nature (tel que BFO que nous avons choisi comme approche pour notre contribution). Sachant que l'objectif est de trouver des approches d'optimisation efficaces qui traitent le problème d'optimisation dynamique, nous verrons que l'application de certaines techniques aux différentes approches naturo-inspirées est la solution la plus adoptée dans la littérature, où un état de l'art sur les différents travaux est présenté dans ce chapitre.

Le quatrième chapitre présente deux contributions. Dans la première contribution, deux algorithmes seront présentés (DBFO et DABFO) ainsi que leurs application sur le problème de la poursuite d'objets. Dans la deuxième contribution nous présenterons l'algorithme BFO multi-population MBFO (Multi Bacterial Foraging Optimization); qui est une approche basée BFO présentée pour la première fois. Nous détaillerons ses principes de fonctionnement, puis nous évaluons ses performances à l'aide du benchmark MPB défini au chapitre 2 et nous finirons par comparer les résultats de ses performances à d'autres résultats de quelques algorithmes de la littérature.

Le chapitre 5 conclut notre travail en discutant les résultats de nos approches tant théorique qu'expérimentale, et en offrant des pistes de recherche qui pourraient être explorées dans le cadre d'autres projets de recherche qui seraient susceptibles de voir le jour au regard de nos conclusions.

Chapitre 2

Optimisation dynamique

2.1 Introduction

L'un des principes les plus fondamentaux de notre monde est la recherche d'un état optimal; cela débute dans le microcosme où les atomes en physique essaient de tisser des liens afin de minimiser l'énergie de leurs électrons. Quand les molécules forment des corps solides au cours du processus de congélation, ils essaient d'assumer des structures cristallines d'énergie optimale. Ces processus bien sûr ne sont guidés par aucun contrôle supervisé mais sont purement des résultats des lois de la physique. La même chose vaut pour le principe biologique de la survie du plus fort avec l'évolution biologique, cela conduit à une meilleure adaptation des espèces à leur environnement. Ici, un optimum local est une espèce bien adaptée qui domine toutes les autres espèces dans ses environs.

Tant que l'humanité existe, la perfection est visée dans de nombreux domaines. Nous voulons atteindre un degré maximal de satisfaction avec le minimum d'effort. Dans notre économie, les profits et les ventes doivent être maximisés et les coûts devraient être aussi faibles que possible. Par conséquent, l'optimisation est l'une des plus anciennes sciences qui s'étend jusque à notre vie quotidienne.

Si quelque chose est importante, générale et assez abstraite, il y a toujours une discipline mathématique qui le traite. L'objectif de l'optimisation globale est de trouver les

meilleurs éléments possibles x^* à partir d'un ensemble X selon un ensemble de critères $F = \{f_1, f_2, \dots, f_n\}$. Ces critères sont exprimés comme des fonctions mathématiques appelées fonctions objectifs.

2.2 Fonction objectif

Une fonction objective $f: \mathbb{X} \longmapsto Y$avec $Y \subseteq R$ est une fonction mathématique qui est soumise à l'optimisation [Thomas 2009]. Le domaine Y d'une fonction objectif ainsi que sa plage doit être un sous-ensemble des nombres réels ($Y \subseteq R$). Le domaine $\mathbb{X}$ de f est appelé espace du problème et peut représenter n'importe quel type d'éléments tels que les nombres, les listes … etc.

f est choisi en fonction du problème à résoudre avec le processus d'optimisation. Les fonctions objectifs ne sont pas nécessairement de simples expressions mathématiques, mais peuvent être des algorithmes complexes qui peuvent par exemple impliquer de multiples simulations. L'optimisation globale comprend toutes les techniques qui peuvent être utilisées pour trouver les meilleurs éléments x^* dans $\mathbb{X}$ par rapport à ces critères $f \in F$.

2.3 Théorie de la complexité

Un problème d'existence peut être transformé en un problème de décision équivalent. Par exemple, le problème du voyageur de commerce qui cherche, dans un graphe dont les arêtes sont étiquetées par des coûts, à trouver un cycle, de coût minimum, passant une fois par chaque sommet, peut s'énoncer en un problème de décision comme suit : Existe-t-il un cycle passant une fois par chaque sommet tel que tout autre cycle passant par tous les sommets ait un coût supérieur de celui-ci. L'équivalence de ces deux problèmes suppose que la démonstration d'existence repose sur un argument constructif, par exemple dans le cas du voyageur de commerce, fournissant effectivement un cycle de coût minimum dans le cas où l'on a montré qu'un cycle de coût minimum existe. Le problème de l'existence d'un cycle de coût minimum est équivalent au problème du voyageur de commerce, au sens où si l'on sait résoudre efficacement l'un, on saura aussi résoudre efficacement l'autre.

La complexité en temps (ou en espace) est définit par le temps (ou l'espace nécessaire) d'exécution d'un algorithme en fonction de la taille du problème, la notation grand-O $O(\dots)$ est souvent utilisée. Un algorithme est dit polynomial si et seulement si sa complexité est bornée par un polynôme $p(n)$, ie. $f(n) = O(p(n))$. Un algorithme est dit exponentiel si sa

complexité ne peut pas être bornée par un polynôme, e.g., $f(n) = a\hat{}n$ avec $a > 1$ une constante. La figure 2.1 décrit les différentes classes.

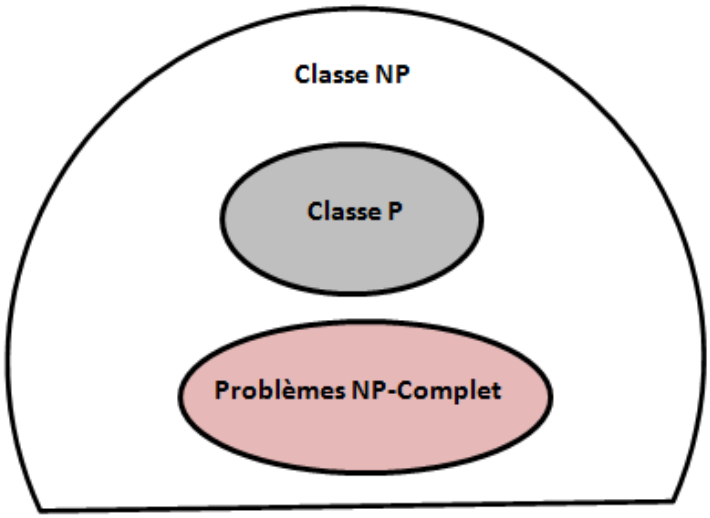

Figure 2.1: Complexité des problèmes.

Classe P : problèmes de décision pour lesquels il existe des algorithmes polynomiaux qui les traitent.

Classe NP : problèmes de décision pour lesquels il existe des algorithmes polynomiaux non déterministes qui les traitent.

Classe NP-complet : un problème de décision A dans NP est NP-complet si tous les autres problèmes de la classe NP se transforment plynômialement dans le problème A.

2.4 Définition formelle de l'optimisation

Pour formaliser le concept d'optimisation on considère X l'ensemble des solutions candidates du problème d'optimisation. Typiquement X est n-dimensionnelle dans certains domaines, par exemple binaire: $X = \{0,1\}^n$ ou des valeurs réelles: $X \subseteq \mathbb{R}^n$. Le domaine X est souvent désigné comme l'espace de recherche. Soit le problème d'optimisation qui est défini par la fonction f appelée la fonction fitness (ou fonction coût, fonction erreur, fonction objectif) et qui définie la qualité des solutions candidates dans X sur un problème donné:

$f: X \longmapsto \mathbb{R}$

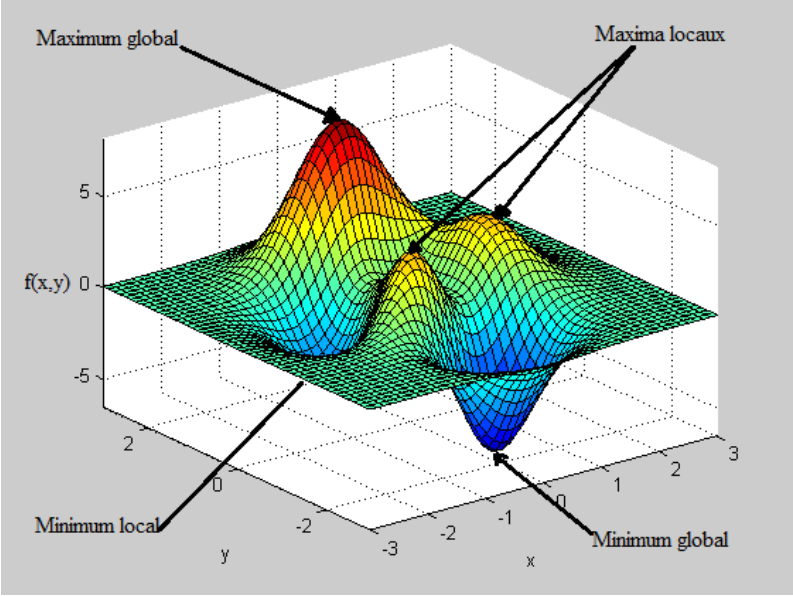

Figure 2.2: Les optima globaux et locaux d'une fonction à deux dimensions.

Un problème de minimisation c'est de minimiser la fonction objectif f et donc obtenir la solution candidate qui minimise f le mieux possible, de sorte que:

$$\forall\, y \in X : f(x) < f(y)$$

Un tel point x est connu comme un minimum global de la fonction f. Il n'est généralement pas possible d'identifier le minimum global exactement dans les solutions d'optimisation. Les solutions candidates ayant suffisamment un bon fitness sont jugées acceptables pour des raisons pratiques.

Les problèmes de maximisation peuvent être optimisés simplement par l'introduction d'une fonction auxiliaire. Supposons que f est une fonction objectif qui doit être maximisée, alors le problème de minimisation analogue peut être considéré simplement en introduisant la fonction : $h(x) = -f(x)$.

2.5 Classification des algorithmes d'optimisation

Dans cette partie, nous allons fournir une classification des algorithmes d'optimisation comme une vue d'ensemble et discuter certains cas d'utilisation de base.

2.5.1 Algorithmes probabilistes et algorithmes déterministes

Figure 2.2 dessine une taxonomie approximative des méthodes d'optimisation globale. En générale, les algorithmes d'optimisation peuvent être divisés en deux catégories de base [Thomas 2009]: algorithmes déterministes et algorithmes probabilistes.

Les algorithmes déterministes sont souvent utilisés que si une relation claire entre les caractéristiques des solutions possibles et de leur utilité existe pour un problème donné, de telle façon, l'espace de recherche peut être efficacement exploré en utilisant par exemple le schéma Diviser Pour Régner. Si la relation entre une solution candidate et son fitness n'est pas si évidente ou trop compliquée, ou la dimension de l'espace de recherche est très grande, ça devient plus difficile de résoudre un problème de manière déterministe. Essayer cela entraînerait une possible énumération exhaustive de l'espace de recherche, ce qui n'est pas pratique, même pour les problèmes relativement petits.

Alors, des algorithmes probabilistes entrent en jeu, les premiers travaux dans ce domaine qui est maintenant devenu l'un des domaines de recherche les plus importants dans l'optimisation ont commencé il ya plus 55 ans. Une famille particulièrement pertinente d'algorithmes probabilistes est les approches Monte Carlo; ils se négocient pour garantir une exactitude de la solution pour une exécution plus courte. Cela ne signifie pas que les résultats obtenus en utilisant ces algorithmes ne sont pas corrects, ils ne peuvent tout simplement être l'optimum global. D'autre part, une solution possible un peu inférieure que la meilleure, est meilleure que celle qui a besoin de 100 ans pour être trouvée.

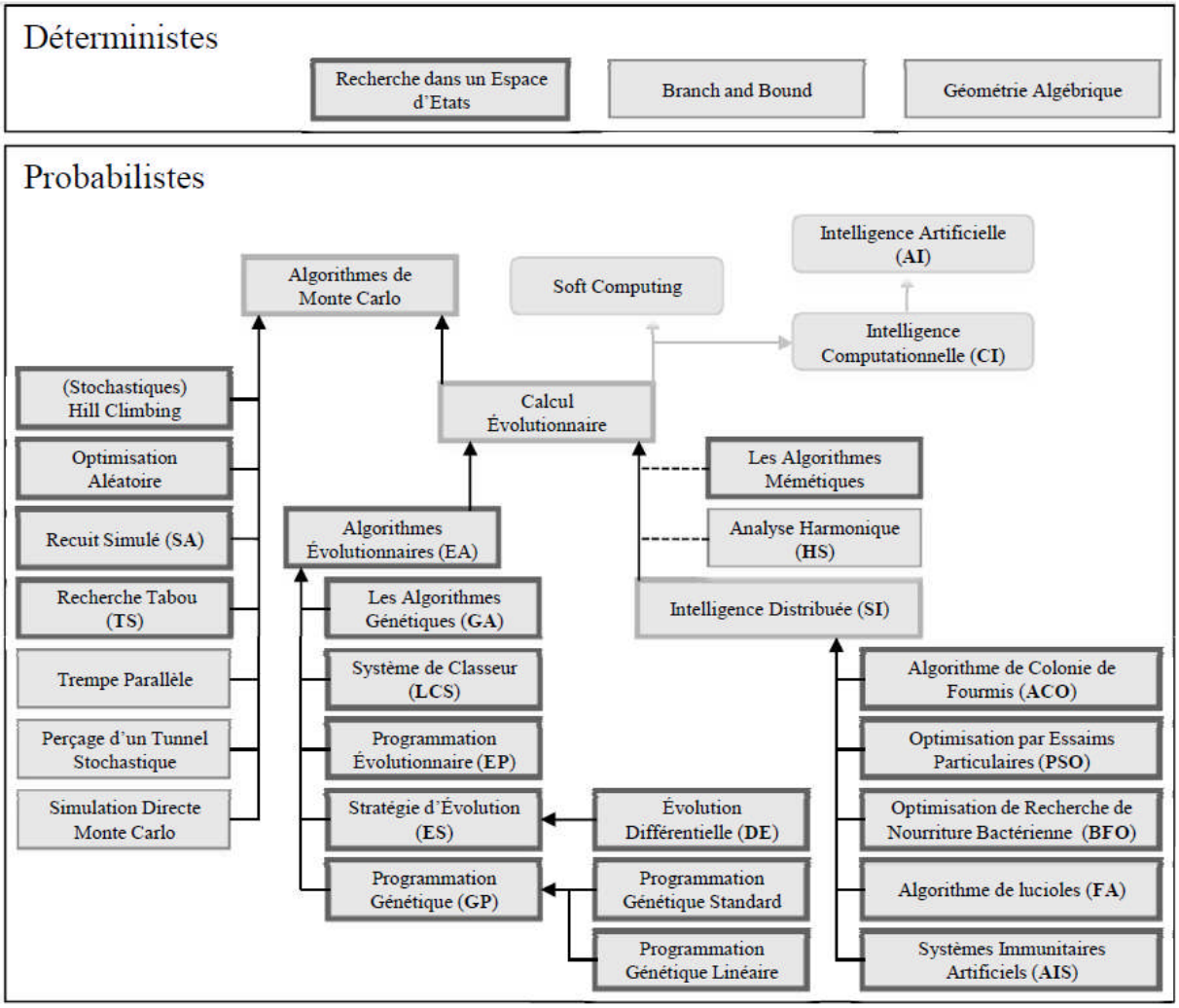

Figure 2.3 : La taxonomie des algorithmes d'optimisation globale.

2.5.2 Algorithmes exactes et Algorithmes approchés

Une heuristique est une méthode de calcul qui fournit rapidement une solution réalisable, pas nécessairement optimale ou exacte pour un problème d'optimisation difficile. Une heuristique diffère d'un algorithme d'approximation dans le sens que même l'heuristique ne garantit pas d'obtenir une solution exacte, elle apporte une garantie quant à la qualité de la solution. Une heuristique utilise les informations actuellement recueillies par un algorithme pour aider à décider quelle est la solution candidate suivante qui doit être testée ou comment le prochain individu peut être produit. Les heuristiques sont généralement dépendantes de la classe du problème. Une méta-heuristique est une méthode pour résoudre des classes de problèmes très génériques, elle combine des fonctions objectifs ou heuristiques de façon

abstraite, généralement sans utiliser des connaissances plus approfondies de leurs structures c'est-à-dire en les traitant comme des procédures de boîte noire.

Les algorithmes déterministes emploient habituellement les heuristiques afin de définir l'ordre de traitement des solutions candidates.

2.6 Quelques problèmes rencontrés dans l'optimisation globale

2.6.1 Convergence prématurée

Un processus d'optimisation est prématurément convergé vers un optimum local s'il n'est plus en mesure d'explorer d'autres parties de l'espace de recherche de la région en cours d'examen et il existe une autre région qui contient une meilleure solution [Thomas 2009]. Figure 2.4 illustre un exemple de la convergence prématurée.

L'existence de plusieurs optima globaux n'est pas une problématique et la découverte d'un seul optimum parmi ces optima peut être encore considérée comme un succès dans de nombreux cas. L'apparition de nombreux optima locaux, cependant, est un cas plus compliqué.

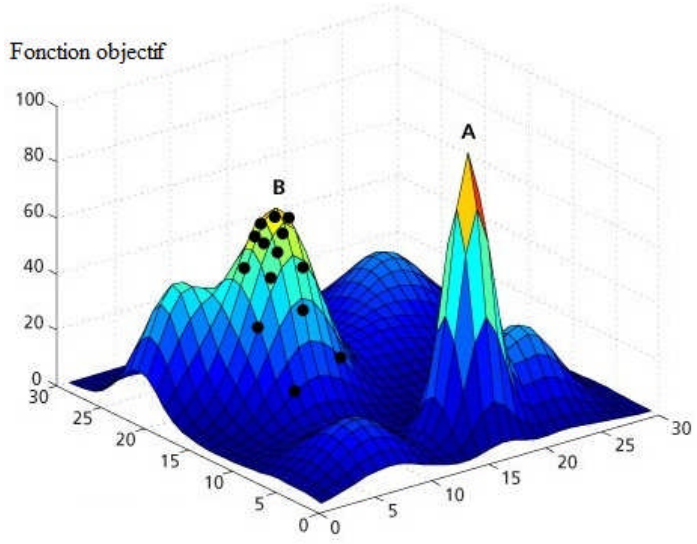

Figure 2.4 : La convergence prématurée dans un espace d'une fonction objectif 2D.

2.6.2 Exploration et exploitation

L'exploration cherche en permanence la totalité de l'espace de recherche pour de meilleures solutions, mais l'exploitation prend les solutions actuellement trouvées et les utilise pour trouver des solutions encore meilleures [Sjoerd 2012]. Équilibrer correctement ces deux mécanismes doit être fait soigneusement et c'est très difficile.

Si les effets de l'exploration sont trop forts, l'algorithme ne converge vers aucune solution, alors que si les effets de l'exploitation sont trop forts, l'algorithme converge trop vite et ne sera pas en mesure d'échapper des optima locaux.

2.6.3 Changement dynamique des problèmes

Beaucoup de problèmes se caractérisent par un paysage dynamique de la fonction objectif qui change avec le temps, comme il est le cas dans notre étude dans ce mémoire. La tâche d'un algorithme d'optimisation est alors de fournir des solutions candidates avec des valeurs objectives momentanément optimales pour chaque point dans le temps. Ici, nous avons le problème que l'optimum dans l'itération t ne sera peut-être pas un optimum dans l'itération t+1.

Le benchmark des pics mobiles (Moving Peaks Benchmark 'MBP') présenté par Branke [Branke 2004] et Morrison [Morrison 1999] est un bon exemple du changement dynamique du paysage de la fonction objectif. Nous trouverons une description détaillée de ce benchmark dans la section 2.12.5.

2.7 Optimisation dynamique

Les télécommunications, les infrastructures, les services et en général tous les aspects relatifs à la société de l'information souffrent d'un ensemble de problèmes complexes où plusieurs objectifs peuvent avoir besoin d'être satisfaits, qui peuvent varier avec le temps, ou lorsque l'incertitude existe dans les valeurs des variables, coefficients, ou même dans les objectifs [Cruz 2011]. En outre, des alternatives qui ont été médiocres dans le passé peuvent être bonnes à l'état actuel ou vice-versa; les critères qui étaient importants avant deviennent non pertinents à l'état actuel et se déplacer dans ces scénarios dynamiques est un défi. Le développement de stratégies de résolution dans le domaine des systèmes intelligents qui peuvent travailler dans tels scénarios dynamiques et imprécis pose de nouveaux défis qui sont abordés de différents points de vue par la communauté scientifique internationale. Beaucoup de problèmes dans ce contexte peuvent être modélisés comme des problèmes d'optimisation dynamiques (DOPs) dans lesquels certains éléments du modèle utilisé peuvent changer au cours de l'optimisation.

En d'autres termes, un DOP est un problème où la fonction objectif ou les contraintes changent avec le temps. La méthode la plus simple pour résoudre ces problèmes c'est d'ignorer le dynamisme, en considérant chaque changement comme l'arrivée d'un nouveau

problème d'optimisation, mais il est souvent impossible, comme il est illustré dans le travail [Branke 2000]. Par conséquent, l'objectif des méthodes pour traiter les DOPs n'est plus de trouver une solution optimale stationnaire, mais aussi de poursuivre son mouvement à travers les espaces de solutions et les espaces de temps aussi prés que possible. Les algorithmes classiques ont été adaptés pour faire face à tels scénarios dynamiques en renforçant leur capacité de poursuivre des optima en mouvement.

2.8 Classification des problèmes de l'optimisation dynamique

Les problèmes d'optimisation dynamique peuvent être classifiés selon les critères suivants:

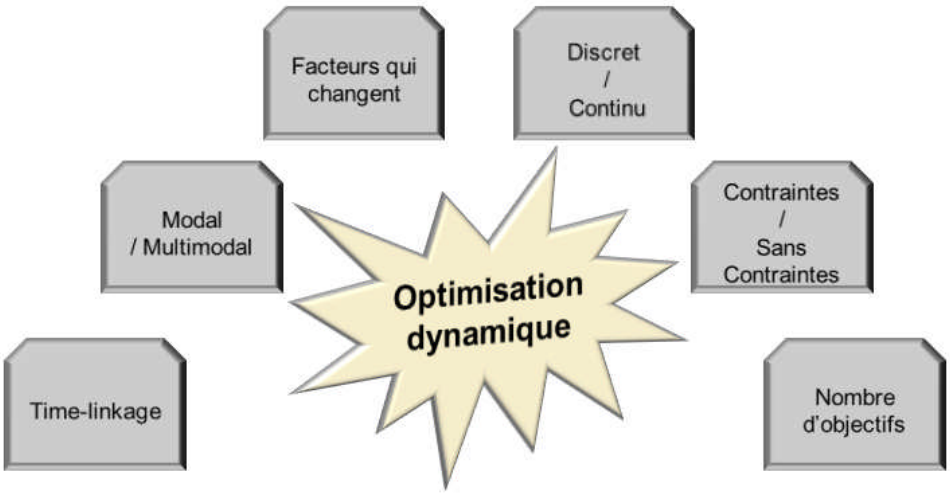

Figure 2.5 : Classification des problèmes de l'optimisation dynamique.

Time linkage : définit si le futur comportement du problème dépend de la solution courante trouvée par l'algorithme ou non [Bosman 2005]; bien que tous les problèmes avec des environnements dynamiques subissent des changements indépendamment des résultats de la recherche, la recherche peut créer d'autres changements dans l'environnement. Pour certains problèmes, les solutions n'ont pas d'effet sur l'environnement. Pour beaucoup d'autres, cependant, la solution spécifique change l'environnement. Par exemple dans le contrôle de la circulation, à la fois l'arrivée de nouveaux véhicules dans le réseau routier et les paramètres d'un signal de trafic déterminent les changements à la fluidité du trafic.

Modal / Multimodal : un problème d'optimisation dynamique peut être modal si l'objectif est de trouver une seule solution. Un problème d'optimisation dynamique est multimodal si l'objectif est de s'intéresser à plusieurs solutions.

Facteurs qui changent : ils peuvent être les paramètres des fonctions objectifs, les domaines des variables, les contraintes ou une combinaison de ces derniers.

Discret / Continu : où les variables qui représentent le problème peuvent être discrètes ou continues.

Avec contraintes / Sans contraintes : Beaucoup de problèmes du monde réels ont un ensemble prédéfini de contraintes qui seront résolues par un algorithme. Ces contraintes peuvent être statiques ou dynamiques qui changent au fils du temps. Un problème dynamique peut être aussi sans contraintes.

Nombre d'objectifs : Les problèmes dynamiques peuvent être mono-objectifs ou multi-objectif.

2.9 Caractéristiques des environnements dynamiques

Une classification des environnements dynamiques est décrite dans [Barlow 2011] selon la fréquence du changement, la prévisibilité du changement, la détectabilité des changements et la répétitivité des environnements :

Fréquence : Dans certains problèmes, les changements peuvent être rares, tandis que dans d'autres, les changements peuvent se produire constamment. L'aspect le plus important de la fréquence de changement est combien de temps un algorithme d'optimisation doit prendre pour trouver une solution à la fois avant qu'il ait un effet sur les performances et avant qu'un autre changement ne se produise.

Gravité : La gravité des changements définie également un problème dynamique. Certains problèmes peuvent avoir des changements qui sont suffisamment petits pour être facilement poursuivies, tandis que d'autres ont de grandes variations discontinues. De petits changements peuvent ne pas avoir un effet important sur le paysage de la fonction objectif, tandis que les grands changements peuvent complètement changer le paysage de la fonction objectif.

Prévisibilité : Dans certains problèmes, des changements suivent un modèle spécifique. Dans d'autres cas, les changements sont complètement aléatoires. Un problème avec des petits changements prévisibles nécessitera un algorithme très différent de celui qui a des changements graves et imprévisibles.

Détectabilité : Dans certains problèmes, les changements dans le paysage de la fonction objectif sont faciles à détecter et on sait exactement quand un changement a eu lieu. Dans

d'autres cas, un certain temps est nécessaire avant qu'il soit clair que l'environnement est changé. Cela peut avoir un effet important sur la façon dont un algorithme résout un problème.

Répétitivité et structure des changements : Certains problèmes peuvent être purement stochastiques, avec des changements complètement aléatoires dans l'environnement. Pour d'autres problèmes, l'environnement dynamique peut faire défiler un nombre fini de configurations distinctes. Pour les problèmes dynamiques les plus intéressants, l'environnement actuel sera similaire aux environnements vus précédemment. Cela peut permettre à des informations du passé d'être utilisées durant la recherche de solutions dans le contexte actuel.

2.10 Définition formelle de l'optimisation dynamique

Les problèmes d'optimisation dynamique (DOPs) peuvent être satisfaits dans de nombreux cas dans le monde réel. Un problème d'optimisation dynamique peut être formulé comme suit:

$$min\, f(x,t)$$

$$h_j(x,t) = 0\ for\ j = 1,2,\dots,u$$

$$g_k(x,t) \leq 0\ for\ k = 1,2,\dots,v$$

Où $f(x,t)$ est la fonction objectif pour un problème de minimisation, $h_j(x,t)$ denote la $j^{ème}$ contrainte d'égalité, et $g_k(x,t)$ dénote la $k^{ème}$ contrainte d'inégalité.

Une façon de faire face au changement consiste à considérer le problème comme nouveau et recommencer à zéro. Toutefois, si le nouvel optimum n'est pas trop loin de l'ancien, en s'appuyant sur les informations actuellement acquises pourrait être utile pour l'adaptation au changement [branke 2000].

Pour cela un algorithme d'optimisation doit non seulement trouver la meilleure solution mais poursuivre cette dernière qui peut changer après qu'elle soit atteinte ou même durant le processus de sa recherche, les changements peuvent se produire fréquemment ou continuellement, généralement dans la plus part des problèmes pratiques le changement est graduel et partiel et pas toujours brusque et total. De ce fait, des techniques adaptatives et efficaces doivent être utilisées pour trouver la nouvelle solution le plus rapidement possible (poursuivre l'optimum).

2.11 Techniques utilisées par l'optimisation dynamique

Plusieurs études ont été faites pour résoudre le problème de l'optimisation dynamique dont la plupart utilisent une ou une combinaison des techniques suivantes :

2.11.1 Introduire la diversité à la détection du changement

Ces méthodes sont généralement utilisées pour réagir à un changement donné dans l'environnement en augmentant la diversité, généralement les différentes approches classiques perdent leurs diversité après la convergence vers un optimum, chose qui pose problème si l'optimum change, cependant la convergence peut devenir préjudiciable.

2.11.2 Maintenir la diversité durant la recherche

Ces méthodes maintiennent la diversité de la population, dans l'espoir que, lorsque la fonction objectif change, la distribution des individus dans l'espace de recherche permettra de trouver rapidement le nouvel optimum.

2.11.3 Utiliser la mémoire

Ces méthodes gardent en mémoire l'évolution des différents optima, pour les utiliser plus tard, ces méthodes sont particulièrement efficaces lorsque l'évolution est périodique, la mémoire peut être intégrée implicitement comme une représentation redondante ou explicitement comme une mémoire séparée, la mémoire implicite stocke des informations du passé comme une partie de l'individu, alors que la mémoire explicite stocke des informations séparément de la population, généralement sous forme d'un ensemble de bonnes solutions précédemment trouvées.

La mémoire explicite a été beaucoup plus largement étudiée et a produit de bien meilleures performances que la mémoire implicite sur les problèmes dynamiques.

Mémoire implicite:

L'utilisation de la diploïdie et la dominance dans les algorithmes génétiques (AG) ont été beaucoup utilisés pour améliorer les performances de l'optimisation dynamique. La diploïdie augmente la diversité dans les AG en permettant aux gènes récessifs de survivre dans une population et de devenir actifs à un moment ultérieur, lorsque le changement de l'environnement les rend plus souhaitables, [Dan 2009].

Mémoire explicite:

La mémoire explicite pour les algorithmes évolutionnaires sauvegarde des informations séparément de la population dans une mémoire externe. Les mémoires explicites ont été très populaires et largement utilisées pour l'optimisation dynamique. D'après Nguyen et al [Nguyen 2012] les méthodes qui utilisent cette approche doivent accomplir quatre taches:

- Quelle est le contenu de la mémoire (directe ou associative) ?
- Comment modifier la mémoire ?
- A quel moment modifier la mémoire ?
- Comment utiliser la mémoire ?

2.11.4 Utiliser la prédiction

Dans certains cas, des changements dans des environnements dynamiques peuvent présenter quelques modèles qui sont prévisibles. Dans ce cas, il pourrait être judicieux d'essayer d'apprendre ces types de modèles à partir de l'expérience de la recherche antérieure et sur la base de ces modèles on tente de prédire les changements à l'avenir.

- prédiction du mouvement de l'optimum qui se déplace (en utilisant une séquence de position trouvées dans le passé).

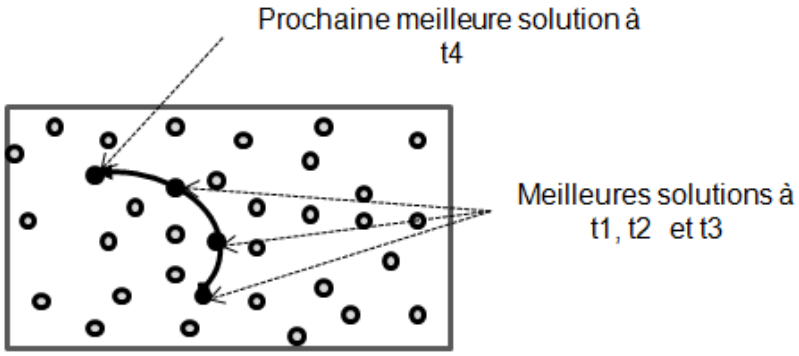

Figure 2.6 : Exemple de la technique de prédiction.

- Prédiction des positions dont les individus doivent être réinitialisés quand un changement arrive.(Ex: En utilisant une fonction de distribution)
- Prédiction du temps dont le changement va se produire et quel environnement possible va apparaitre dans le changement

2.11.5 Utiliser plusieurs populations

La division de la population en plusieurs sous-populations permet de poursuivre plusieurs pics dans la fonction fitness. Les différentes sous-populations conservent alors des informations sur plusieurs régions prometteuses de l'espace de recherche, ainsi augmenter la probabilité de trouver de nouveaux optima et garder une bonne diversité. Dans [Nguyen 2012], cette méthode doit atteindre deux objectifs: l'attribution de différentes tâches à des sous-populations et la division des sous-populations et faire en sorte que les sous-populations ne se chevauchent pas pour éviter la situation où plusieurs sous-populations poursuivent le même pic.

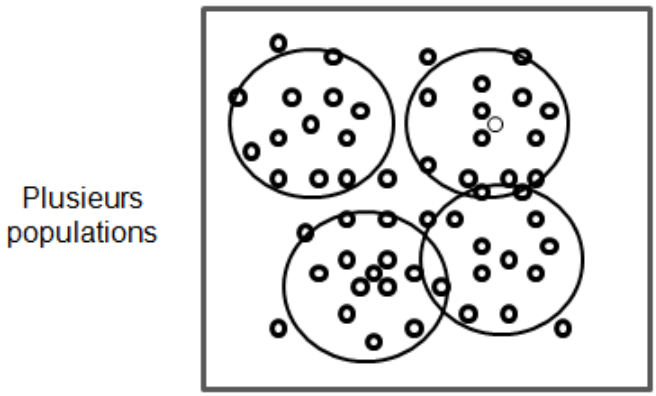

Figure 2.7 : Exemple de la technique multi-population.

2.12 Quelques problèmes d'optimisation dynamique

2.12.1 Problème dynamique du voyageur de commerce D-TSP

Parmi les problèmes les plus célèbres et connus dans le domaine d'optimisation le problème du voyageur de commerce (TSP). Il consiste à trouver la plus courte tournée permettant de visiter toutes les villes et de revenir au point de départ en ne visitant chaque ville qu'une seule fois. L'objectif est de minimiser le coût total prévu, c'est-à-dire la somme des distances utilisées pour visiter toutes les villes.

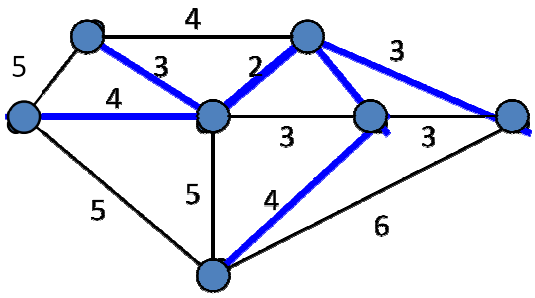

Figure 2.8 : Problème du TSP.

Deux modèles de DTSP différentes ont été décrits dans la littérature [Guntsch, 2001] [Eyckelhof 2002], le premier consiste à insérer ou supprimer les villes dans une instance de problème donnée, le deuxième modèle est appliqué pour décrire les embouteillages entre les villes en changeant les valeurs des arcs reliant les villes.

2.12.2 Problème dynamique du sac à dos D-KNAPSACK

Le problème du sac à dos est un problème d'optimisation classique avec de nombreuses applications pratiques : plusieurs objets qui ont des capacités et des valeurs connues doivent être emballées dans un sac à dos de capacité connue de telle façon à maximiser la valeur totale des objets inclus.

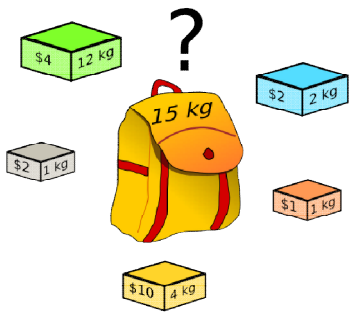

Figure 2.9 : Problème du sac à dos.

Dans la version dynamique et stochastique proposée par Ross et Tsang (1889), les objets arrivent séquentiellement dans le temps et leur combinaison poids-valeurs est stochastique et sera connue par les concepteurs à l'heure d'arrivée [Deniz 2011].

2.12.3 Problème dynamique de coloration de graphe D-GC

C'est un problème appartenant aux problèmes NP-Complet grâce à la difficulté trouvée lors de sa résolution, il consiste à attribuer une couleur à chacun de ses sommets de

manière à ce que deux sommets reliés par une arête soient de couleurs différentes. Il est souvent recherché d'utiliser un nombre minimal de couleurs, dit nombre chromatique.

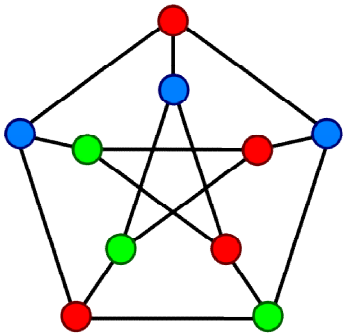

Figure 2.10 : Problème de coloration de graphe.

Ça version dynamique consiste à insérer ou supprimer des sommets ou des arêtes au cours du temps dans une instance de problème donnée [Ouerfelli 2011]

2.12.4 Problème dynamique de satisfiabilité maximale (D-MAX-SAT)

Le problème de satisfiabilité (SAT) est un type des problèmes de satisfaction de contraintes (CSP). Dans SAT, les variables binaires qui définissent un problème peuvent prendre que deux valeurs 0 ou 1, qui correspondent à "Vrai" et "Faux", respectivement [Pinto 2013]. Les formules logiques sont construites avec ces variables binaires et le problème est défini pour trouver une affectation des variables qui se traduisent par "Vrai" pour chacune des formules logique. MAX-SAT est l'extension du problème d'optimisation connexe au problème SAT. Dans MAX-SAT, le but est de trouver une solution qui minimise le nombre de formules violées.

Dans un problème MAX-SAT dynamique, les clauses peuvent être ajoutées ou retirées d'une instance au fil du temps. Cela implique un modèle d'un système qui est soumis à des contraintes différentes à différents moments dans le temps. Une autre façon de définir le problème dynamique est de garder le même ensemble fixe de clauses, mais permettre certaines variables d'etre modifiées à Vrai ou Faux à différents points dans le temps.

2.12.5 Problème du MPB (Moving Peaks Benchmark)

Dans le but de combler l'écart entre les problèmes du monde réel très complexes et difficiles à comprendre et à tous les problèmes trop simples, [Branke 1999] suggère un problème avec un paysage de la fonction objectif multidimensionnel composé de plusieurs

pics, dont la hauteur, la largeur et la position de chaque pic est légèrement modifié à chaque fois qu'un changement se produit dans l'environnement.

C’est à noter qu'un petit changement dans le paysage de la fonction objectif du problème peut avoir deux conséquences: il peut parfois être suffisant pour adapter la solution actuelle pour atteindre le nouvel optimum, et parfois, il peut être nécessaire de passer à un autre, déjà légèrement inférieur mais qui a une meilleure solution à ce moment. Le premier se produit par exemple lorsque l'optimum est légèrement décalé, le second se produit lorsque la hauteur des pics change tel qu'un pic différent devient un pic maximum. Dans ces cas, un algorithme d’optimisation doit essentiellement traverser une vallée, pour atteindre le nouveau pic maximum.

Une fonction de test à n dimensions et m pics peut être formulée comme suit:

$$\boldsymbol{F(\vec{x}, t) = \max\left(B(\vec{x}), \max_{i=1\ldots m} P(\vec{x}, h_i(t), w_i(t), \vec{p}_i(t))\right)}$$

où $\boldsymbol{B(\vec{x})}$ est le paysage basique de la fonction objectif qui est invariable dans le temps, et $\boldsymbol{P}$ est la fonction définissant une forme de pic, où chacun des $\boldsymbol{m}$ pics a ses propre paramètres variants dans le temps : l’hauteur $\boldsymbol{(h)}$, la largeur $\boldsymbol{(w)}$ et l’emplacement $\boldsymbol{\vec{p}}$.

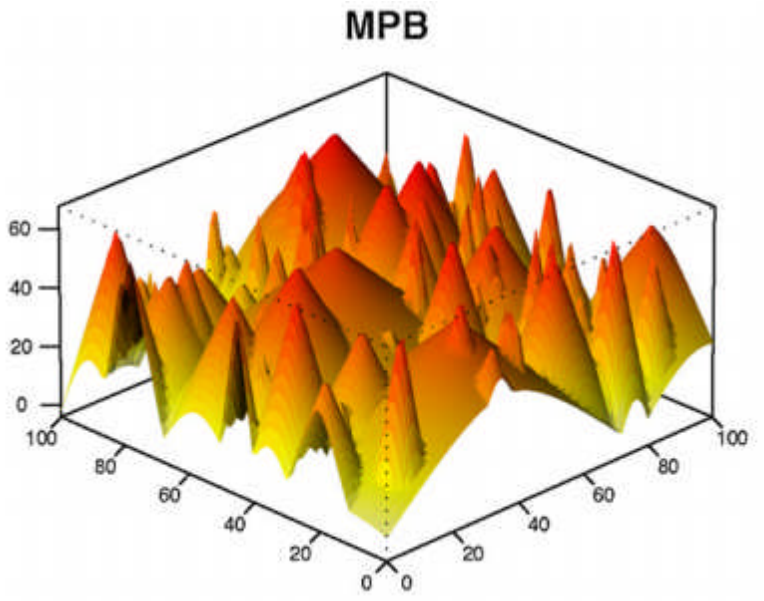

Figure 2.11 : Problème du MPB.

Les coordonnées, la hauteur et la largeur de chaque pic sont initialisées en fonction d'un générateur de nombres aléatoires intégré.

Ensuite, chaque $\boldsymbol{\Delta_e}$ évaluations, la hauteur et la largeur de chaque pic sont modifiées par l'ajout d'une variable aléatoire gaussienne. L'emplacement de chaque pic $\boldsymbol{i}$ est déplacé par un vecteur $\vec{\boldsymbol{v}}_{\boldsymbol{i}}$ de longueur fixe $\boldsymbol{s}$ dans une direction aléatoire (pour $\boldsymbol{\lambda = 0}$) ou dans une direction en fonction de la direction précédente (pour $\boldsymbol{\lambda} > 0$).

Dans l'ensemble, le paramètre $\boldsymbol{s}$ permet de contrôler la gravité d'un changement, $\boldsymbol{\Delta_e}$ déterminera la fréquence des changements, $\boldsymbol{\lambda}$ permet de contrôler si les changements présentent une tendance.

Plus formellement, un changement d'un seul pic peut être décrit comme:

$$\boldsymbol{\sigma} \in \boldsymbol{N(0,1)}$$

$$\boldsymbol{h_i(t) = h_i(t-1) + heigth_{severity} \cdot \sigma}$$

$$\boldsymbol{w_i(t) = w_i(t-1) + width_{severity} \cdot \sigma}$$

$$\vec{\boldsymbol{p}}_{\boldsymbol{i}}\boldsymbol{(t) = }\vec{\boldsymbol{p}}_{\boldsymbol{i}}\boldsymbol{(t-1) + }\vec{\boldsymbol{v}}_{\boldsymbol{i}}\boldsymbol{(t)}$$

Le vecteur de déplacement $\vec{\boldsymbol{v}}_{\boldsymbol{i}}$ est une combinaison linéaire d'un vecteur aléatoire $\vec{\boldsymbol{r}}$ et le vecteur de déplacement précédent $\vec{\boldsymbol{v}}_{\boldsymbol{i}}\boldsymbol{(t-1)}$, et il est normalisé à la longueur $\boldsymbol{s}$

$$\vec{\boldsymbol{v}}_{\boldsymbol{i}}\boldsymbol{(t)} = \frac{\boldsymbol{s}}{|\,\vec{\boldsymbol{r}} + \vec{\boldsymbol{v}}_{\boldsymbol{i}}\boldsymbol{(t-1)}|}\,(\boldsymbol{(1-\lambda)}\vec{\boldsymbol{r}} + \boldsymbol{\lambda}\vec{\boldsymbol{v}}_{\boldsymbol{i}}\boldsymbol{(t-1)})$$

Le vecteur aléatoire $\vec{\boldsymbol{r}}$ est créé en tirant des nombres aléatoires pour chaque dimension et en normalisant sa longueur à $\boldsymbol{s}$.

La complexité de la fonction peut facilement être adaptée par l'augmentation du nombre de dimensions et / ou le nombre de pics, ou en utilisant des fonctions de pics plus complexes.

2.13 Conclusion

La majorité des approches inspirées de la nature utilisées dans l'optimisation statique ne fonctionnent pas bien dans des environnements dynamiques dans leur forme conventionnelle, cependant, les bonnes approches pour l'optimisation dynamique doivent être capables de suivre les optima au fil du temps. Ils doivent également être en mesure de maintenir la diversité de la population afin d'éviter la convergence de la population en arrivant à une stagnation dans des optima locaux. Plusieurs techniques ont été proposées dans la littérature pour faire améliorer ces approches face au dynamisme des environnements. Ces

technique peuvent être résumées en : la réactivité aux changements, le maintien de la diversité, l'utilisation de la mémoire, l'utilisation de la prédiction et l'utilisation de plusieurs populations.

Chapitre 3

Approches naturo-inspirées pour l’optimisation dynamique

3.1 Introduction

La nature a été souvent utilisée comme source d'inspiration pour les techniques de recherche pour résoudre des problèmes d’optimisation continue et discrète. L'un des aspects clés de ces systèmes est la capacité de s'adapter aux conditions environnementales qui changent. Pourtant les techniques d’optimisation naturo-inspirées sont principalement conçues pour résoudre les problèmes statiques (problèmes qui ne changent pas lors de leur résolution) plutôt que leurs homologues dynamiques [Hendtlass 2006]. Ceci est principalement dû au fait que l'incorporation du contrôle de la recherche temporelle est une tâche difficile. Récemment, toutefois, un plus grand ensemble de travaux ont été réalisés sur des versions améliorées de ces méta-heuristiques inspirées de la nature, tels que les algorithmes génétiques, l'optimisation par colonie de fourmis et l'optimisation par essaim particulaire afin de leurs permettre de résoudre des problèmes d'optimisation dynamique.

3.2 Quelques approches inspirées de la nature

La nature est une grande et immense source d'inspiration pour résoudre des problèmes difficiles et complexes de l'informatique car elle présente dextrement de divers phénomènes dynamiques, robustes, complexes et fascinants. Elle trouve toujours la solution optimale pour résoudre son problème en maintenant l'équilibre parfait entre ses composantes [Lin 2007]. C'est la poussée derrière l’optimisation naturo-inspirée. Les algorithmes inspirés de la nature sont des méta-heuristiques qui imitent la nature pour résoudre des problèmes d'optimisation ouvrant une nouvelle ère dans le calcul. Pour les dernières décennies, de nombreux efforts de recherche ont été concentrés dans ce domaine particulier. Tout en étant récents, les résultats sont très étonnants, ce qui a élargit la portée et la viabilité des algorithmes inspirés de la

nature d'explorer de nouveaux domaines d'application et plus de possibilités en matière d'informatique.

Les paragraphes suivants présentent un aperçu général de quelques algorithmes d'optimisation inspirés de la nature basés sur l'évolution (GA, GP, ES, EP, DE) et de quelques algorithmes d'optimisation inspirés de la nature basés sur l'intelligence distribuée (PSO, ACO, BFO, FA, AIS).

3.2.1 Les approches basées sur les algorithmes évolutionnaires

Les algorithmes évolutionnaires (EA) sont des méta-heuristiques d'optimisation basées population. Les solutions candidates d'un problème d'optimisation sont définies comme des individus d'une population. L'évolution de la population conduit à trouver de meilleures solutions. Les individus sont évalués dans l'environnement et certains mécanismes inspirés de l'évolution biologique sont appliqués pour l'évolution de la population [Eisuke 2011].

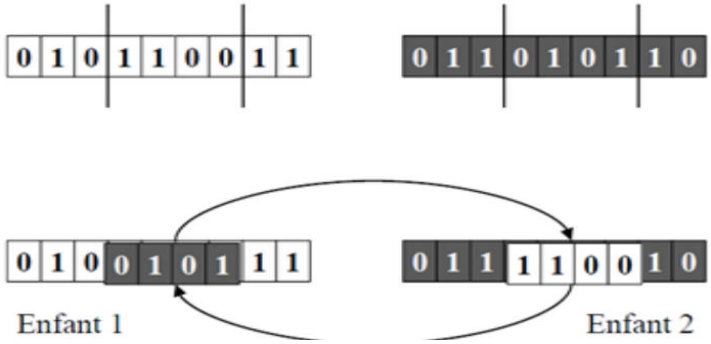

Figure 3.1: Exemple du croisement.

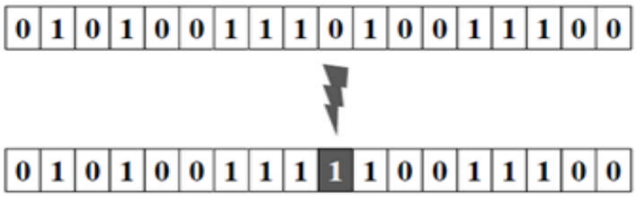

Figure 3.2 : Exemple de la mutation.

Les algorithmes génétiques (GA), les stratégies d'évolution (ES), la programmation génétique (GP), la programmation évolutionnaire (EP) et l'évolution différentielle sont des algorithmes évolutionnaires très populaires [Barlow 2011], une description de ces algorithmes est donnée dans ce qui suit.

3.2.1.1 Les algorithmes génétiques GA

L'objectif des algorithmes génétiques est de maximiser le profit des solutions candidates dans une population contre une fonction de coût du domaine du problème. La stratégie des algorithmes génétiques est d'employer les mécanismes de croisement et de mutation génétiques sur les solutions candidates de la population, où la fonction objectif appliquée à une représentation décodée d'un candidat gère les contributions probabilistes, ce qui permet à une solution candidate donnée d'apporter à la génération suivante de nouvelles solutions candidates [Brownlee 2011].

3.2.1.2 Programmation génétique GP

L'objectif de la programmation génétique consiste à utiliser l'induction pour mettre au point un programme informatique. Ce résultat est obtenu en utilisant des opérateurs évolutifs sur des programmes candidats avec une structure arborescente afin d'améliorer l'ajustement adaptatif entre la population des programmes candidats et une fonction d'objectif. Une évaluation d'une solution candidate implique son exécution [Brownlee 2011].

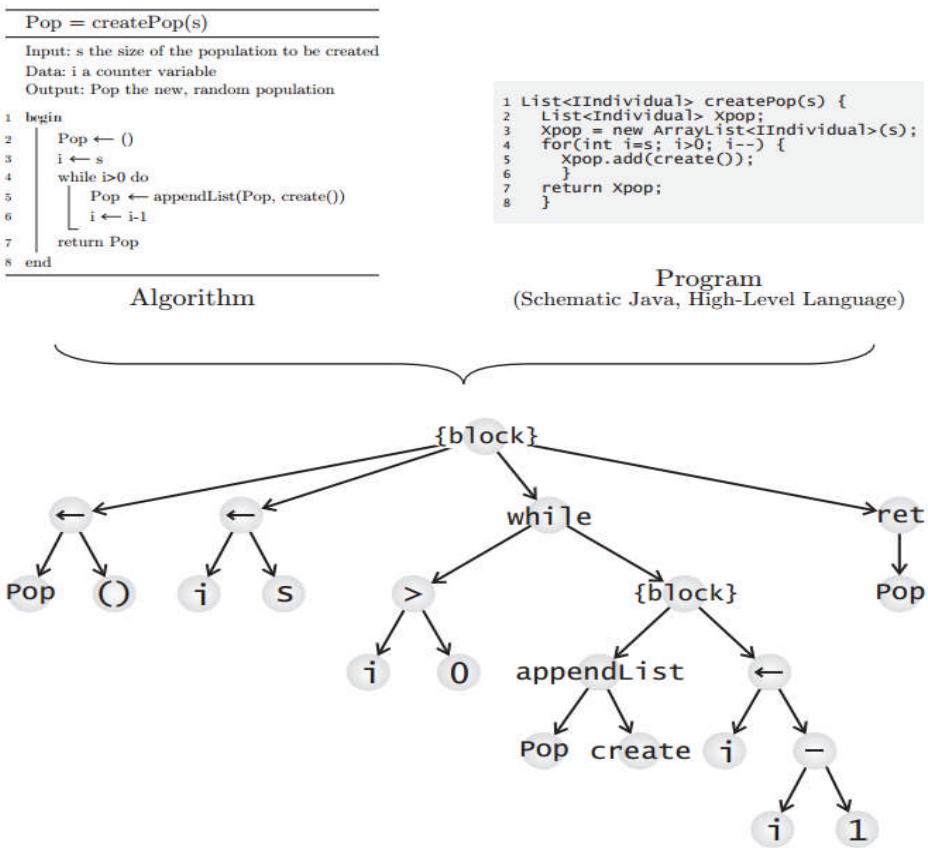

Figure 3.3 : Représentation abstraite de l'arbre syntaxique des algorithmes / programmes [Weise 2009].

3.2.1.3 Stratégies d'évolution ES

Les stratégies d'évolution (ES) sont les premiers types d'algorithmes évolutionnaire qui ont été utilisés. Elles ont été introduites par Rechenberg en 1965 comme des heuristiques d'optimisation basées sur le principe de l'adaptation et de l'évolution. Les ES utilisent usuellement des vecteurs de nombres réels pour représenter l'espace des solutions. La mutation et la sélection représentent les opérateurs fondamentaux de la reproduction. Le croisement est moins utilisé. Le schéma d'exécution générale des ES est semblable au fonctionnement de base d'un algorithme évolutionnaire [Weise 2009].

3.2.1.4 Programmation évolutionnaire EP

La programmation évolutionnaire est différente des autres principaux types d'algorithmes évolutionnaires déjà introduits. Il n'existe pas de définition claire ou une variante algorithmique pour la programmation évolutionnaire. Il ya cependant une différence sémantique; alors que seules les individus d'une espèce sont la métaphore biologique pour les solutions candidates dans les autres algorithmes évolutionnaires, dans la programmation évolutionnaire, une solution candidate est considérée comme une espèce elle-même. Ainsi, la mutation et la sélection sont les seuls opérateurs utilisés dans EP et les croisements ne sont généralement pas appliqués. Le schéma de sélection utilisé dans la programmation évolutionnaire est assez similaire à la méthode ($\mu + \lambda$) dans les stratégies d'évolution [Weise 2009].

3.2.1.5 Évolution différentielle DE

L'évolution différentielle est une méthode de recherche directe et parallèle qui utilise NP vecteurs de paramètres de dimension D comme une population pour chaque génération G.

$$x_{i,G}, i = 1,2, \dots, NP$$

NP ne change pas au cours du processus de minimisation. Les vecteurs de la population initiale sont choisis au hasard et doivent couvrir l'espace de recherche entier. En règle générale, une distribution de probabilité uniforme est supposée pour toutes les décisions aléatoires. Dans le cas où une solution préliminaire est disponible, la population initiale peut être générée par l'addition des écarts aléatoires d'une distribution normale à la solution symbolique. L'évolution différentielle génère de nouveaux vecteurs de paramètres en ajoutant la différence pondérée entre deux vecteurs de la population à un troisième vecteur, cette opération est appelée Mutation [Storn 1997].

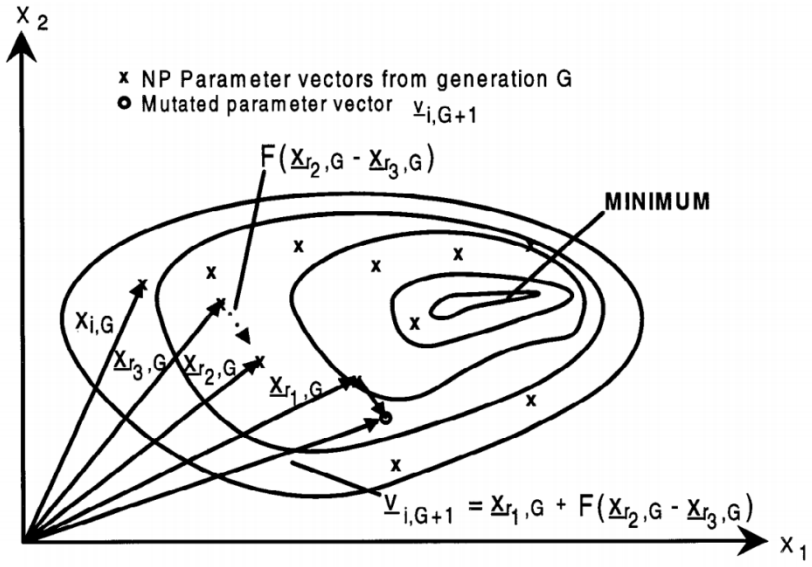

Figure 3.4 : Un exemple d'une fonction objectif à 2-dimensions qui montre ses contours et le processus de génération du vecteur mutant [Storn 1997].

Les paramètres du vecteur muté sont ensuite mélangés avec les paramètres d'un autre vecteur prédéterminé (le vecteur cible) pour donner le vecteur d'essai. Le mélange des paramètres est souvent désigné comme "croisement". Si le vecteur d'essai donne une valeur de fonction de coût plus faible que le vecteur cible, le vecteur d'essai remplace le vecteur cible à la génération suivante. Cette dernière opération est appelée "Sélection". Plus précisément la stratégie de base de DE peut être décrite comme suit:

- **La mutation** :

Pour chaque vecteur cible $x_{i,G}, i = 1,2, \dots, NP$, un vecteur mutant est généré conformément à

$$v_{i,G+1} = x_{r_1,G} + F.(x_{r_2,G} - x_{r_3,G})$$

avec des indexes aléatoires $r_1, r_2, r_3 \in \{1,2, \dots, NP\}$, entiers, mutuellement différents et $F > 0$. Les nombres entiers choisis de façon aléatoire r_1, r_2 et r_3 sont également choisis de façon à être différents de l'indice courant i, de sorte que NP doit être supérieur ou égal à quatre pour que cette condition soit permise. F est un facteur constant et réel $\in [0, 2]$ qui contrôle l'amplification de la variation différentielle $(x_{r_2,G} - x_{r_3,G})$. La figure 3.4 montre un exemple bidimensionnel qui illustre les différents vecteurs qui jouent un rôle dans la génération de $v_{i,G+1}$.

- **Le croisement** :

Afin d'accroître la diversité des vecteurs des paramètres perturbés, le croisement est introduit. A cette fin, le vecteur d'essai:

$$u_{i,G+1} = (u_{1i,G+1}, u_{2i,G+1}, \dots, u_{Di,G+1})$$

est formé, où

$$u_{ji,G+1} = \begin{cases} v_{ji,G+1}\ if\ (randb(j) \le CR)\ or\ j = rnbr(i) \\ u_{ji,G}\ \ if\ (randb(j) > CR)\ and\ j \neq rnbr(i) \end{cases}$$

$$J = 1,2, \dots, D$$

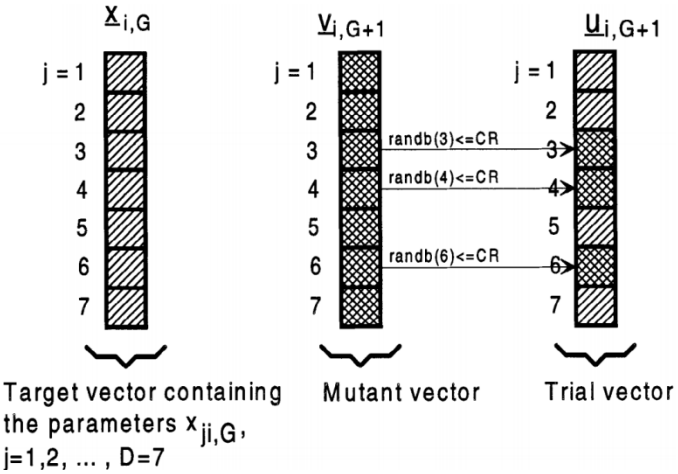

Figure 3.5 : Illustration du processus de croisement pour d = 7 paramètres [Storn 1997].

$randb(j)$ est la j[ème] évaluation d'un générateur de nombres aléatoires uniforme avec un résultat ∈ [0,1], CR est la constante de croisement ∈ [0,1] qui doit être déterminée par l'utilisateur, $rnbr(i)$ est un indice choisi aléatoirement ∈ 1,2,…,D qui garantit que $u_{ji,G+1}$ utilise au moins un paramètre de $v_{i,G+1}$. La figure 3.5 donne un exemple du mécanisme de croisement des vecteurs à 7-dimensions.

- **La sélection** :

Pour décider si le vecteur doit devenir un membre de la génération G+1 ou non, le vecteur d'essai $u_{i,G+1}$ est comparé au vecteur cible $x_{i,G}$. Si le vecteur $u_{i,G+1}$ donne une valeur de la fonction de coût inférieure à $x_{i,G}$, $x_{i,G+1}$ est remplacé par $u_{i,G+1}$, sinon l'ancienne valeur $x_{i,G}$ est conservée.

3.2.1.6 Etat de l'art sur les approches évolutionnaires dynamiques

Dans [COBB 1990] H. G. Cobb a utilisé une méthode appelée l'Hyper-mutation basée sur les algorithmes génétiques, où il propose d'augmenter le taux de mutation lors d'une détection du changement de l'environnement.

Dans [Vavak 1998] F. Vavak et al proposent l'opérateur VLS ('Variable local search') qui permet une recherche locale autour des emplacements actuels dans l'espace de recherche représenté par les chromosomes avant qu'un changement de l'environnement ait lieu. La portée de la recherche est variable et peut être progressivement étendue pour correspondre à l'ampleur des changements de l'environnement. L'opérateur VLS est déclenché lorsque la moyenne des performances optimales de la population dans le temps est inférieure à un seuil prédéfini.

Dans [Riekert 2009] Riekert et Malan ont proposé 'Adaptive Genetic Programming' qui non seulement augmente le taux de la mutation, mais aussi réduit l'élitisme et augmente le taux de la probabilité du croisement après qu'un changement soit détecté.

L'idée d'introduire la diversité après la détection d'un changement a également été utilisée dans l'optimisation multi-objectif dynamique. Par exemple, dans un algorithme multi-population pour DMO [Goh 2009], quand un changement est détecté, quelques individus choisis au hasard et certains individus concurrents provenant d'autres sous-populations sont introduits à chaque sous-population pour augmenter la diversité.

Afin de favoriser une exploration continue de l'espace de recherche [Grefenstette 1992], John J. Grefenstette propose d'augmenter la version standard du GA comme suit: Après l'étape de la mutation, on insère une étape d'hyper-mutation partielle, qui remplace un pourcentage de la population par des individus générés aléatoirement. Le pourcentage remplacé est appelé le taux de remplacement. L'effet recherché est de maintenir un niveau constant de l'exploration de l'espace de recherche, tout en essayant de réduire au minimum la perturbation de la recherche en cours.

Dans TDGA (Thermodynamical Genetic Algorithm) [Mori 1998], l'opération de la sélection est conçue pour minimiser l'énergie libre de la population, où l'énergie libre F est définie par: F = (E) - HT,

La minimisation de l'énergie libre peut être considérée comme étant un équilibre entre la minimisation de la fonction d'énergie et le maintient de la diversité mesuré par l'entropie.

Les individus ayant des valeurs d'énergie relativement faibles (ou relativement des bons fitness) seront affectés par des priorités pour survivre à la prochaine génération. En même temps, les individus ayant des gènes rares seront également préservés en raison de leurs contributions à la minimisation de l'énergie libre en augmentant le terme de l'entropie HT. Ainsi, la diversité de la population peut être contrôlée en ajustant la température T de manière explicite.

Dans [Morrison 2003] Ronald W. Morrison voulai améliorer la capacité de recherche de la population initiale en étalant les membres individuels à travers l'espace de recherche de sorte que leurs positionnements maximisent la probabilité de détection d'une caractéristique d'intérêt de l'espace de recherche.

L'algorithme PBIL [Baluja 1994] utilisé aussi par Shengxiang Yang et al [Yang 2005], a comme but de générer un vecteur de probabilité réel estimé, lorsqu'il est échantillonné, il crée des solutions de haute qualité avec une forte probabilité. PBIL commence à partir d'un vecteur de probabilité initial avec des valeurs de chaque entrée fixée à 0,5 ce qui signifie lors de l'échantillonnage par ce vecteur de probabilité initial, des solutions aléatoires seront crées. Cependant, que la recherche progresse, les valeurs du vecteur de probabilité sont graduellement apprises vers des valeurs représentant des solutions d'évaluation élevées. Le processus d'évolution est décrit comme suit :

À chaque itération, un ensemble d'échantillons (solutions) est créé en fonction du vecteur de probabilité courant. L'ensemble d'échantillons est évalué en fonction de la fonction fitness du problème spécifique. Puis le vecteur de probabilité est appris (poussé) vers des solutions avec de fitness très élevé. La distance du vecteur de probabilité poussé dépend du paramètre du taux d'apprentissage. Après que le vecteur de probabilité est mis à jour, un nouvel ensemble de solutions est généré par échantillonnage à partir d'un nouveau vecteur de probabilité et ce cycle est répété. Comme la recherche progresse, les entrées du vecteur de probabilité s'éloignent de leur paramètre initial de 0,5 à 0,0 ou à 1,0.

Le modèle EEGA (the Estimation of Evolvability Genetic Algorithm) [Wang 2006] présenté par Yao Wang et Mark Wineberg en 2006 est composé de trois sous-populations interdépendantes. La première sous-population est le composant principal, qui accueille la majorité des individus de la population et qui est choisie en fonction du fitness tout comme cela se fait dans un AG standard. La seconde sous-population est constituée d'individus sélectionnés pour une augmentation de leurs fitness par rapport à leurs parents, dont leurs

propriétés d'évolution offrent des bénéfices immédiats. En d'autres termes cette sous-population est choisie selon le changement du fitness. La troisième sous-population récompense les membres qui ressemblent le moins à leurs parents.

Dans [Bui 2005] Lam T. Bui et al proposent d'utiliser deux objectifs pour résoudre un problème d'optimisation mono-objectif. Le premier objectif est toujours considéré comme l'objectif unique de l'optimisation du problème, tandis que le deuxième objectif est artificiel, conçu pour maintenir la diversité.

Dans [Parrott 2006], Iason Hatzakis et David Wallace proposent une approche qui utilise un équilibre entre la convergence et la diversité ainsi que la stratégie de prédiction d'anticipation afin de faciliter la découverte du nouvel optimum dans le cas où la prévision est sans succès, cette dernière stratégie proposée dans ce travail consiste à utiliser l'historique du chemin de l'optimum dans le temps pour prévoir l'emplacement de l'optimum dans de la prochaine étape. La prévision pour l'emplacement à côté de l'optimum est utilisée pour créer un ensemble d'individus (consistant dans le cas le plus simple d'un seul individu qui se trouve sur les coordonnées de prédiction) qui est appelé l'ensemble de prédiction. Dès que la prochaine étape arrive et la fonction objectif change, l'ensemble de prédiction est inséré dans la population de l'algorithme.

Dans [Rossi 2008], Claudio Rossi et al proposent une prédiction qui fournie la future position qui represente la nouvelle position de l'optimum. Trois techniques peuvent être mises au point qui permettent l'ajout de ce type d'information dans un algorithme évolutionnaire.

- **Opérateurs génétiques modifiés:** Cette technique consiste en une conception spécialisée des opérateurs de mutation et de croisement.

- **Fonction objectif modifiée:** des informations heuristiques supplémentaires peuvent être intégrées dans la fonction objectif sous forme de termes supplémentaires, appelées fonctions de raffinage.

- **Individus doués:** Cette technique consiste à créer de nouveaux individus en utilisant les informations provenant de la prédiction, et de les insérer directement dans la population. Si la prédiction était assez précise, le nouvel optimum sera parmi ces insertions.

Dans [Zhou 2007] Aimin Zhou et al ont étudié comment générer une population initiale proche du front de Pareto changé (dans l'optimisation dynamique multi-objectif) après qu'un changement soit détecté. Ils ont construit des modèles de prédiction pour prédire

l'emplacement du nouvel ensemble de Pareto basé sur les informations collectées de la recherche précédente. Une fois qu'un changement est détecté, on prévoie les nouveaux emplacements d'un certain nombre de solutions Pareto dans l'espace de décision. Les individus de la population initiale du problème changé sont générés autour de ces points prévus. De cette façon, l'ensemble de Pareto et le front de Pareto changés peuvent être trouvés plus efficacement par l'algorithme.

Dans [Simoes 2008] Anabela Simoes et Ernesto Costa utilisent une méthode qui implique l'utilisation d'une mémoire pour les derniers individus qui ont un bon fitness, en plus de la population normale. Cette mémoire s'enchaîne avec deux autres modules: l'un basé sur la régression linéaire et l'autre soutenu par les chaînes de Markov. La régression linéaire est utilisée pour estimer quand le prochain changement de l'environnement va se passer. Les chaînes de Markov sont utilisées pour modéliser ce qui est connu à propos de tous les environnements possibles et les transitions entre ces environnements. La chaîne de Markov est utilisée pour prédire les nouveaux environnements qui vont très probablement apparaître dans le futur. La sortie du module de la régression linéaire est basée sur le temps des changements passés. Une fois ce moment défini, le modèle de chaînes de Markov est utilisé pour prédire à quoi vont ressembler les nouveaux environnements possibles.

Dans [Oppacher 1999], Franz Oppacher et Mark Wineberg ont défini SBGA (Shifting Balance Genetic Algorithm) qui est un algorithme génétique multi-population semblable au modèle Island (île). Mais contrairement aux autres modèles Island, il ne divise pas la population en plusieurs dèmes avec la même taille. En SBGA les populations sont classées en deux catégories: une large population centrale appelée le cœur et une série de plus petites populations appelées les colonies. Le cœur est responsable d'explorer la région de l'espace de recherche qui semble la plus prometteuse et performante en exploitation, en recevant les immigrants élites des différentes colonies. Tandis que les colonies explorent l'espace de recherche où la population cœur n'avait pas marqué encore une présence. Deux mécanismes sont présentés :

- Sélection de la distance à partir du cœur :

Le SBGA dispose d'un mécanisme permettant de déterminer si un individu de la colonie est entrain de chercher dans la même région que le cœur; qui est implémenté comme une distance par rapport au cœur; Il s'agit de la distance de Hamming moyenne entre un élément d'une colonie et chaque élément du cœur. La distance est utilisée comme une

fonction-objectif pour les membres de la colonie lorsque la colonie devient trop proche. Les membres seront ensuite sélectionnés pour la reproduction, non seulement en fonction de leurs valeurs de la fonction-objectif mais aussi en fonction de leurs distances par rapport au cœur. Où Ils vont évoluer dans une nouvelle zone de l'espace de recherche.

- Migration à partir d'une colonie vers le cœur:

Dans le SBGA, la colonie envoie des membres appelés les migrants vers le cœur. Durant la migration, la colonie peut envoyer tous ses membres au cœur ou seulement une partie de celle-ci. Les migrants sont choisis parmi les membres élites de la colonie.

Le nombre de générations entre les migrations est appelé l'intervalle de migration.

Ng et Wong proposent un schéma diploïde avec quatre allèles possibles (0 et 1, chacune dominante et récessive). Pour être en mesure de s'adapter rapidement aux changements, ils suggèrent d'utiliser un mécanisme de changement de dominantes dans lequel toutes les paires d'allèles sont inversés chaque fois que le fitness d'un individu diminue de plus de 20%. Dans les expériences présentées, le schéma diploïde surpasse le schéma haploïde ainsi que le schéma triallelic de Goldberg et Smith (où un allèle peut prendre une des trois valeurs "0", "1 récessif" et "1 dominant" testé sur un problème du sac à dos dynamique, qui rapporte de meilleurs résultats par rapport à un simple GA.

Dans une utilisation précoce de la mémoire, Ramsey et Grefenstette [Ramsey 1993] ont créé une mémoire basée-cas qui enregistre les dernières bonnes solutions ainsi que des informations sur les environnements où ces solutions ont été créées. Quand un changement est détecté, les entrées avec des environnements étroitement correspondants sont trouvées et les solutions de ces entrées sont utilisées pour réinitialiser une partie de la population. Le problème dynamique est épisodique, pour que les entrées de la mémoire soient enregistrées une fois par période. La mémoire est autorisée à accroître sans limite et n'est jamais réduite en taille.

Dans ''Memory Enhanced EA'', Jurgen Branke [Branke 1999] suggère de diviser la population en deux (une population-mémoire et une population-recherche). La première population basée-mémoire est responsable de se souvenir des bonnes anciennes solutions, maintenir un minimum de qualité et initier les sauts. L'autre population recherche constamment de nouveaux sommets et les soumet à la mémoire. Pour faire valoir

l'exploration, cette deuxième population est réinitialisée au hasard après la détection de chaque changement dans l'environnement.

Récemment Xingguang Peng et al [Peng 2010] ont proposé les estimations des algorithmes de distribution (EDA). La distribution de probabilité conjointe aux solutions de hautes performances est présentée par un modèle de probabilité. Cela signifie que les zones prioritaires de l'espace de recherche de solutions sont caractérisées par le modèle de probabilité. De ce point de vue, un environnement basé-identification du schéma de gestion de la mémoire (EI-MMS) est proposé pour adapter l'EDA codé-binaire afin de résoudre des problèmes d'optimisation dynamique (DOPs). Dans ce schéma, les modèles probabilistes qui caractérisent l'espace de recherche des environnements dynamiques sont stockés et récupérés afin d'adapter les EDAs en fonction des changements de l'environnement.

Lili LIU et al [LIU 2012] ont proposé un model Voisin-Proche (Near-Neighbor) et un système immunitaire basé sur l'algorithme d'évolution différentielle (NIDE) pour viser les problèmes d'optimisation dynamique. La motivation fondamentale derrière NIDE est de permettre aux individus de rechercher les différentes régions prometteuses pour promouvoir la diversité de la population et de récupérer les informations mémorisées pour détecter et répondre efficacement aux changements. Trois techniques ont été incorporées dans l'algorithme NIDE pour améliorer son adaptabilité dans des environnements dynamiques. La première stratégie comprend le schéma de la mémoire voisine et le voisin proche, qui vise à fournir un guidage droit pour chaque individu à la recherche du pic à proximité ainsi d'orienter l'individu vers une direction indiquée dérivé du vecteur différentiel. La deuxième stratégie est le schéma du système d'amplification multidirectionnelle liée-différenciation (DRMA). Ce schéma utilise un vecteur plutôt qu'une valeur scalaire pour améliorer la capacité d'exploration des individus, ce qui encourage les individus à poursuivre les optima dans des environnements dynamiques. La troisième stratégie utilise les mécanismes basés système immunitaire, ce qui devrait maintenir la diversité de la population ainsi de permettre la recherche dans des régions prometteuses et donc produisant un meilleur suivi dans des environnements dynamiques.

3.2.3 Les approches basées sur l'intelligence distribuée

Figure 3.6 : Intelligence distribuée

L'intelligence distribuée (Swarm Intelligence 'SI') est une métaphore de calcul et de comportement pour résoudre les problèmes distribués inspirés des exemples de la nature fournis par des insectes sociaux comme les fourmis, les termites, les abeilles, les guêpes et aussi par les essaims, les troupeaux et les phénomènes chez les groupes des vertébrés tels que les groupes de poissons et les groupes d'oiseaux. En d'autres termes, l'intelligence distribuée est basée sur les principes qui sous-tendent le comportement des systèmes naturels constitués de plusieurs agents et en exploitant les formes des communications locales et le contrôle hautement distribué. Ainsi, l'approche SI constitue un modèle très pratique et puissant qui a beaucoup simplifié la conception de solutions distribuées aux différents types de problèmes. Au cours des dernières années, l'intelligence distribuée a été appliquée avec succès à une grande série d'applications, y compris les algorithmes d'optimisation, les réseaux de communication et la robotique [Olariu 2006].

Une étude de l'approche Intelligence Distribuée révèle un ensemble de principes d'organisation utiles qui peuvent guider la conception d'applications distribuées efficaces pour plusieurs types de problèmes. L'Intelligence Distribuée possède les caractéristiques suivantes:

Autonomie: Le système ne nécessite pas de contrôle ou d'entretien extérieur. Les individus sont autonomes, qui contrôlent leur propre comportement.

Adaptabilité: Les interactions entre les individus peuvent se produire à travers la communication directe ou indirecte à travers l'environnement local; deux individus interagissent indirectement lorsque l'un d'eux modifie l'environnement et l'autre répond au nouvel environnement à une date ultérieure. En exploitant ces formes de communications locales. Les individus ont la capacité de détecter des changements dans l'environnement dynamique. Ils peuvent ensuite adapter de manière autonome leur propre comportement à ces nouveaux changements.

Evolutivité: Les capacités de l'Intelligence Distribuée peuvent être utilisées en utilisant des groupes composés de quelques-uns à des milliers d'individus ayant la même architecture de contrôle.

Flexibilité: Aucun individu de l'essaim n'est essentiel. Un individu peut être dynamiquement ajouté, enlevé ou remplacé.

Robustesse: l'Intelligence Distribuée fournit un bon exemple d'une architecture fortement distribuée qui améliore considérablement la robustesse. Il n'existe pas de coordination centrale, ce qui signifie qu'il n'ya pas de point de défaillance unique. Le système d'essaim permet la redondance, ce qui est essentiel pour la robustesse.

Massivement parallèle: le système d'essaim est massivement parallèle et son fonctionnement est réellement distribué. Les tâches accomplies par chaque individu au sein de son groupe sont les mêmes. Si on considère chaque individu comme une unité de traitement, l'architecture 'Intelligence Distribuée' peut être considérée comme une architecture SIMD.

L'auto-organisation: les systèmes d'essaim mettent l'accent sur les capacités d'auto-organisation. L'intelligence exposé n'est pas présente chez les individus, mais émerge plutôt d'une certaine manière sur l'ensemble de l'essaim. En d'autres termes, si on considère chaque individu comme une unité de traitement, des solutions aux problèmes obtenus ne sont pas prédéfinies ou préprogrammées mais elles sont déterminées collectivement à la suite du programme exécuté.

Rentabilité: Le système de type essaim consiste en un ensemble fini d'agents homogènes dont chacun a assez de capacités limitées sur lui même. En outre, chaque agent possède les mêmes capacités et le même algorithme de contrôle. Il est clair que l'autonomie et le contrôle hautement distribué offerts par le modèle d'essaim simplifient beaucoup la tâche de concevoir l'implémentation des algorithmes parallèles et du matériel.

3.2.3.1 Optimisation par essaims particulaires PSO

Kennedy et Eberhart [Eberhart 1995] ont développé l'algorithme PSO en tenant compte du comportement des essaims dans la nature comme les oiseaux, les poissons … etc. PSO combine l'auto-expérience avec l'expérience sociale. Dans cet algorithme, une solution candidate est présentée comme une particule. PSO utilise un ensemble de particules volantes (des solutions qui changent) dans une zone de recherche (solutions actuelles et possibles) ainsi que le mouvement vers une zone prometteuse pour arriver à un optimum global [Sedighizadeh 2009].

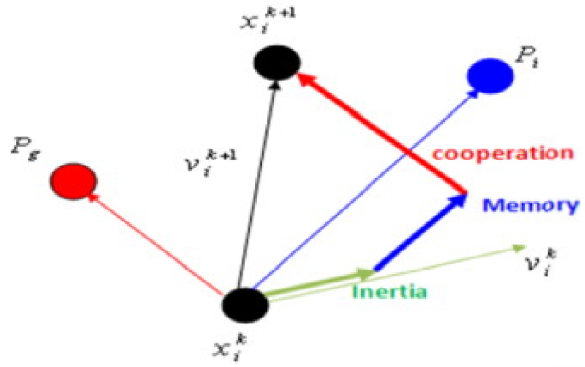

Figure 3.7 : Description de la vitesse et des mises à jour des positions dans l'optimisation par PSO pour un espace de paramètres à deux dimensions [Hassan 2004].

$$prtvel_j^i = \chi\left(w \times prtvel_j^{i-1} + c_1 \times rand \times \left(pbest_j^{i-1} - prtpos_j^{i-1}\right)\right) +,$$

$$c_2 \times rand \times (gbest^{i-1} - prtpos_j^{i-1})$$

$$\chi = \frac{2}{\left|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}\right|}, \varphi = \varphi_1 + \varphi_2, \varphi W > 4$$

$$prtpos_j^i = prtpos_j^{i-1} + prtvel_j^i$$

Où,

c_1, c_2 : Les facteurs d'équilibre entre l'effet de la connaissance de soi et la connaissance sociale dans le déplacement de la particule en direction de la cible. Habituellement, la valeur 2 est suggérée pour les deux facteurs dans la littérature.

rand : Un nombre aléatoire entre 0 et 1, qui est différent à chaque itération

w : La masse d'inertie

$pbest_j^i$: La meilleure position d'une particule

$gbest_j^i$: La meilleure position au sein de l'essaim

$prtvel_j^i$: La vitesse de la particule j dans la ième itération

$prtpos_j^i$: La position de la particule j dans la ième itération

Pseudo-code d'un PSO basique:

Générer aléatoirement une population initiale
Répéter
    Calculer les valeurs du fitness des particules
    Modifier les meilleures particules dans l'essaim
    Choisir la meilleure particule
    Calculer les vitesses des particules
    Mettre à jour les positions des particules
Jusqu'à critère d'arrêt

3.2.3.2 Algorithme de colonie de fourmis ACO

L'optimisation par colonie de fourmis est une méta-heuristique dans laquelle les fourmis artificielles d'une colonie coopèrent pour trouver de bonnes solutions à des problèmes d'optimisation difficiles. La coopération est un élément clé pour la conception des algorithmes ACO: Le choix est d'allouer les ressources de calcul à un ensemble d'agents relativement simples (fourmis artificielles) qui se communiquent indirectement par la Stigmergie (c'est par la médiation de la communication indirecte assurée par l'environnement). Les bonnes solutions sont une propriété émergente de l'interaction coopérative des agents. Une fourmi artificielle ACO est une procédure constructive stochastique qui construit progressivement une solution en ajoutant des composants de solutions opportunément définies à une solution partielle en cours de construction [Dorigo 2003].

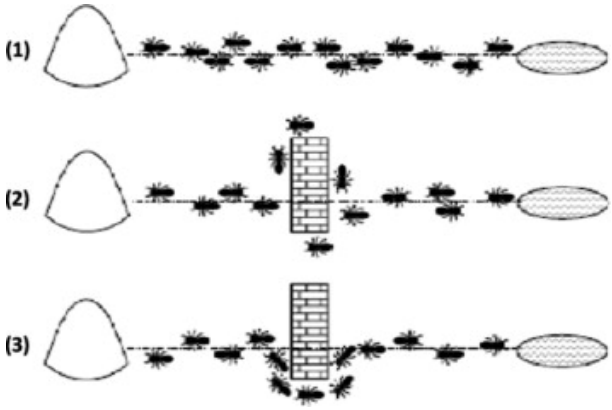

Figure 3.8 : Recherche des fourmis du plus court chemin de la nourriture vers le nid en suivant la phéromone.

D'une façon informelle, un algorithme ACO peut être imaginé comme l'interaction de trois procédures: *Construire-Solutions-Fourmis*, *Mise-à-jour-Phéromones* et *Opérations-Démons*.

Algorithme de base ACO

ConstruireSolutionsFourmis
Mise-à-jourPhéromones
OpérationsDémons / optionnel

End

- Procédure ConstruireSolutionsFourmis : gère une colonie de fourmis qui visitent concurremment et de manière asynchrone les états adjacents du problème considéré en se déplaçant dans les nœuds voisins du graphe du problème. Les fourmis se déplacent en appliquant une politique de décision locale stochastique qui fait utiliser la phéromone et de l'information heuristique. De cette manière, les fourmis construisent progressivement des solutions au problème d'optimisation. Une fois une fourmi a construit une solution, ou alors que la solution est en cours de construction, la fourmi évalue la solution (partielle) qui sera utilisée par la procédure *MiseàjourPhéromones* pour décider quelle est la quantité de la phéromone à déposer.

- Procédure MiseàjourPhéromones : est le processus par lequel les traces de la phéromone sont modifiées. La valeur des sentiers peut soit augmentée (comme dans le cas où les fourmis déposent de la phéromone sur les connexions qu'ils utilisent), ou diminuée due à l'évaporation de la phéromone. D'un point de vue pratique, le dépôt du phéromone augmente la probabilité que ces connexions qui ont été soit utilisés par de nombreuses fourmis ou par au moins une fourmi et qui produisent une très bonne solution soient utilisés à nouveau par les futures

fourmis. Différemment, l'évaporation de la phéromone met en œuvre une forme utile de l'oubli; elle permet d'éviter la convergence trop rapide de l'algorithme vers une région sous-optimale en favorisant ainsi l'exploration de nouvelles zones de l'espace de recherche.

- Procédure OpérationsDémons : Cette procédure est utilisée pour mettre en œuvre des actions centralisées qui ne peuvent être exécutées par les fourmis simples. Des exemples d'actions démons sont l'activation d'une procédure d'optimisation locale ou la collecte d'une information globale qui peut être utilisée pour décider s'il est utile ou non de déposer de la phéromone supplémentaire pour pousser le processus de recherche dans une perspective non local. Comme exemple pratique, le démon peut observer le trajet trouvé par chaque fourmi dans la colonie et sélectionne une ou quelques fourmis (par exemple, ceux qui ont construit les meilleures solutions dans les itérations de l'algorithme) qui sont alors autorisés à déposer des phéromones supplémentaires sur les composants qu'ils ont utilisé.

3.2.3.3 Optimisation par recherche de nourriture bactérienne BFO

L'optimisation par recherche de nourriture bactérienne est inspirée par le sabot exposé par des comportements de recherche de nourriture bactérienne [Tang 2006]. Les bactéries ont tendance à se rassembler dans les zones riches en éléments nutritifs par une activité appelée "*Chimiotaxie*". Il est connu que les bactéries nagent en tournant les flagelles actionnés par un moteur réversible, incorporé dans la paroi de la cellule (E. coli a 8 à 10 flagelles placées de façon aléatoire sur le corps de cellule). Quand tous les flagelles tournent dans le sens antihoraire, ils forment un compact propulsant hélicoïdalement la cellule suivant une trajectoire hélicoïdale, qui est appelé "Courir" (Run). Lorsque les flagelles tournent dans le sens horaire, ils tirent tous sur la bactérie dans des directions différentes, ce qui fait culbuter la bactérie (Tumble) [Passino 2002].

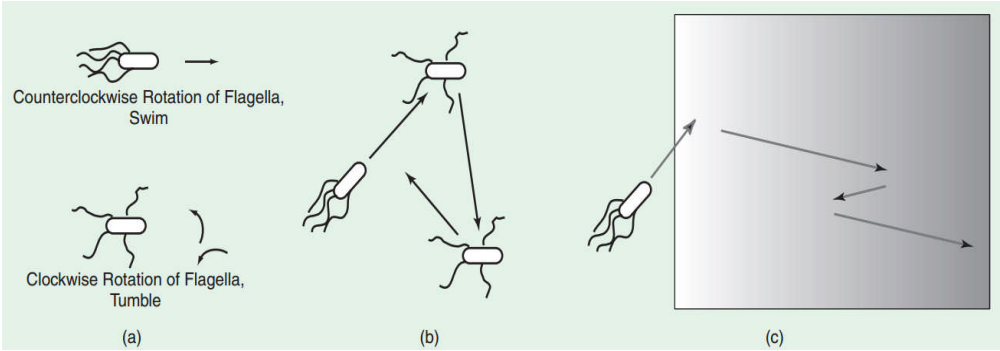

Figure 3.9 : La nage, les culbutes et le comportement Chimiotaxie de l'E. coli. [Passino 2002].

- Chimiotaxie

La Chimiotaxie bactérienne est basée sur la suppression des culbutes dans les cellules qui se produisent par hasard pour se déplacer dans une direction donnée. Les bactéries font des décisions en fonction de leur milieu ambiant. Le mouvement de chaque bactérie flagellée peut être décrit en termes d'intervalles de course dans lequel la cellule nage en ligne droite entrecoupée par des culbutes lorsque l'organisme subit une réorientation aléatoire.

Dans l'algorithme BFO existant, une unité de marche avec une direction aléatoire représente une *Culbute* (*Tumble*) et une unité de marche dans le même sens que la dernière étape indique un *Courir* (*Run)* comme il est indiqué dans la Figure 3.9. Après une étape de déplacement, la position de la i^{ème} bactérie peut être représentée comme suit :

$$\theta_i(j+1,r,l) = \theta_i(j,r,l) + C(i) * \phi(j)$$

où $\theta_i(j+1,r,l)$ indique la position de la i^{ème} bactérie à la j^{ème} étape chimiotactique dans la r^{ème} étape de reproduction au l^{ème} événement de l'élimination et de dispersion. $C(i)$ est la longueur d'une unité de marche qui est réglée pour être une constante. $\phi(j)$ est l'angle de direction de la j^{ème} étape, lorsque son activité est Run, $\phi(j)$ est la même que $\phi(j-1)$, sinon, $\phi(j)$ est un angle aléatoire généré dans la plage $[0,2\,\pi]$.

Avec l'activité de *Run* ou *Tumble* prise à chaque étape du processus *Chimiotaxie*, le fitness noté $J_i(j,r,l)$ sera calculé.

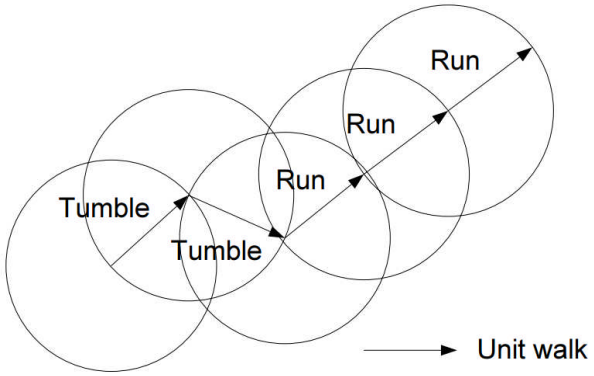

Figure 3.10 : Exemple du processus de la Chimiotaxie [Tang 2006].

- La reproduction

Le fitness total de chaque bactérie est calculé comme la somme des fitness au cours de sa vie, $\sum_{j=1}^{Nc} J_i(j,r,l)$ qui est obtenu après toutes les étapes de la Chimiotaxie, où Nc est le pas maximum dans un processus de Chimiotaxie. Toutes les bactéries sont classées dans

l'ordre inverse en fonction de leurs fitness. Dans l'étape de reproduction, seule la première moitié de la population survit et une bactérie survivante se divise en deux bactéries identiques, qui occupent les mêmes positions dans l'environnement à la 1ère étape. Ainsi, la population des bactéries reste constante dans chaque processus de Chimiotaxie.

- La dispersion et l'élimination

La Chimiotaxie fournit une base pour une recherche locale, le processus de reproduction accélère la convergence. Alors que dans une grande partie la Chimiotaxie et la reproduction ne sont pas assez suffisantes pour la recherche globale des optima. Comme les bactéries peuvent se coincer dans les positions initiales ou dans les optima locaux, il est possible que la diversité des algorithmes BFO change graduellement ou brusquement pour éliminer des accidents d'être piégé dans les optima locaux. Dans BFO, l'événement de dispersion se passe après un certain nombre de processus de reproduction. Une bactérie est choisie selon une probabilité P_{ed} prédéfinie pour être dispersée et déplacée à une autre position dans l'environnement. Ces événements peuvent empêcher efficacement de tomber dans les optima locaux, mais de façon inattendue perturbent le processus d'optimisation.

Le coût des bactéries est réduit par son interaction avec d'autres cellules. Cette fonction interaction g est calculée comme suit:

$$g(cell_k) = \sum_{i=1}^{S}\left[-d_{attr} \times \exp\left(-w_{attr} \times \sum_{m=1}^{P}\left(cell_m^k - other_m^i\right)^2\right)\right]$$

$$+ \sum_{i=1}^{S}\left[d_{repel} \times \exp\left(-w_{repel} \times \sum_{m=1}^{P}\left(cell_m^k - other_m^i\right)^2\right)\right]$$

Où $cell_k$ est une cellule donnée, d_{attr} et w_{attr} sont les coefficients d'attraction, h_{repel} et w_{repel} sont les coefficients de répulsion, S est le nombre de cellules dans la population, P est le nombre de dimensions d'un vecteur de position donné.

Les paramètres restants de l'algorithme sont comme suit $Cells_{num}$ est le nombre de cellules maintenues dans la population, N_{ed} est le nombre d'étapes d'élimination et de dispersion, N_{re} est le nombre d'étapes de la reproduction, N_c est le nombre d'étapes de la chemotaxie, N_s est le nombre d'étapes de la nage pour une cellule donnée, $Step_{size}$ est un vecteur de directions aléatoires avec le même nombre de dimensions que l'espace de

problème et chaque valeur $\in [-1, 1]$ et P_{ed} est la probabilité pour qu'une cellule soit l'objet d'une élimination et dispersion.

- Pseudo-code du BFO

Input: $Problem_{size}$, $Cells_{num}$, N_{ed}, N_{re}, N_c, N_s, $Step_{size}$, $d_{attract}$, $w_{attract}$, $h_{repellant}$, $w_{repellant}$, P_{ed}

Output: $Cell_{best}$

```
Population ← InitializePopulation(Cells_num, Problem_size);
for l = 0 to N_ed do
    for k = 0 to N_re do
        for j = 0 to N_c do
            ChemotaxisAndSwim(Population, Problem_size, Cells_num, N_s, Step_size, d_attract,
            w_attract, h_repellant, w_repellant);
            foreach Cell ∈ Population do
                if Cost(Cell) ≤ Cost(Cell_best) then
                    Cell_best ← Cell;
                end
            end
        end
        SortByCellHealth(Population);
        Selected ← SelectByCellHealth(Population, Cells_num);
        Population ← Selected+ Selected;
    end
    foreach Cell ∈ Population do
        if Rand() ≤ P_ed then
            Cell ← CreateCellAtRandomLocation();
        end
    end
end
return Cell_best;
```

Pseudo-code de la fonction ChemotaxisAndSwim:

Input: Population, $Problem_{size}$, $Cells_{num}$, N_s, $Step_{size}$, $d_{attract}$, $w_{attract}$, $h_{repellant}$, $w_{repellant}$

foreach *Cell* ∈ Population do

  $Cell_{fitness}$ ← Cost(*Cell*) + Interaction(*Cell*, Population, $d_{attract}$, $w_{attract}$, $h_{repellant}$, $w_{repellant}$);

  $Cell_{health}$ ← $Cell_{fitness}$;

  Cell'← ∅;

  for *i* = 0 to N_s do

    RandomStepDirection ← CreateStep($Problem_{size}$);

    Cell'←TakeStep(RandomStepDirection, $Step_{size}$);

    $Cell'_{fitness}$←Cost(*Cell'*)+Interaction(*Cell'*, Population, $d_{attract}$, $w_{attract}$, $h_{repellant}$, $w_{repellant}$);

    if $Cell'_{fitness}$ > $Cell_{fitness}$ then

      $i \leftarrow N_s$;

    else

      Cell ← *Cell'*;

      $Cell_{health}$← $Cell_{health}$+ $Cell'_{fitness}$;

    end

  end

end

3.2.3.4 Algorithme des lucioles FA

Le clignotement des lucioles est un spectacle étonnant dans le ciel d'été dans les régions tropicales et tempérées. Il ya environ deux mille espèces de lucioles et la plupart des lucioles produisent des clignotements courts et rythmés. Les modèles de flashs sont souvent uniques pour une espèce particulière. La lumière clignotante est produite par un processus de bioluminescence et les véritables fonctions de ces systèmes de signalisation sont encore à débattre [Yang 2010]. Toutefois, deux fonctions fondamentales de ces flashes sont : attirer des partenaires d'accouplement (communication) et attirer les proies potentielles. En outre, le clignotement peut également servir comme un mécanisme d'alerte et de protection. Les femelles répondent à un modèle unique de clignotement d'un mâle de sa même espèce, alors que dans certaines espèces comme les Photuris, les lucioles femelles peuvent imiter le modèle

de clignotement d'accouplement des autres espèces afin d'attirer et de manger les lucioles mâles qui pourraient le confondre avec un potentiel compagnon approprié.

Le clignotement peut être formulé d'une telle manière qu'il soit associé à la fonction objectif à optimisée, ce qui rend facile et possible la formulation de nouveaux algorithmes d'optimisation.

Certaines caractéristiques du clignotement des lucioles peuvent être idéalisées afin de développer des algorithmes inspirés-lucioles. Pour plus de simplicité dans la description de l'algorithme des lucioles (FA), les trois règles suivantes sont utilisées:

1) Toutes les lucioles sont unisexes, de telle sorte qu'une luciole sera attirée par d'autres lucioles indépendamment de leur sexe.

2) L'attractivité est proportionnelle à leur luminosité, ainsi, étant donné deux lucioles clignotantes, la moins lumineuse se déplacera vers la plus lumineuse. L'attractivité est proportionnelle à la luminosité et tous les deux diminuent en tant que leur distance augmente. S'il n'y a pas une luciole plus lumineuse qu'une luciole particulière, elle se déplace au hasard.

3) La luminosité d'une luciole est affectée ou déterminée par la fonction objectif.

Dans l'algorithme des lucioles, il existe deux questions importantes: La variation de la lumière en intensité et la formulation de l'attractivité. Pour simplifier, il est supposer que l'attraction d'une luciole est déterminée par sa luminosité qui à son tour est associée à la fonction d'objectif.

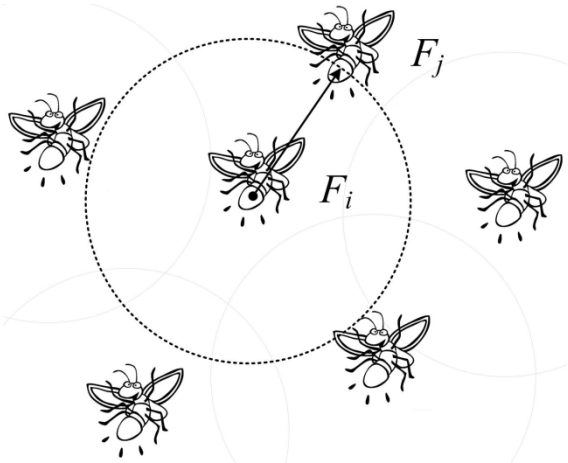

Figure 3.11 : Attractivité des lucioles par la lumière.

Dans le cas le plus simple pour les problèmes d'optimisation, la luminosité I d'une luciole à un endroit particulier x peut être choisi comme $I(x)\ \alpha\ f(x)$. Cependant,

l'attractivité β est relative, elle devrait être vue dans les yeux du spectateur ou jugée par les autres lucioles. Ainsi, elle varie en fonction de la distance r_{ij} entre la luciole i et la luciole j. En outre, l'intensité de la luminosité diminue avec la distance de sa source et la lumière est également absorbée par les médias de sorte qu'il sera permis à l'attractivité d'être variée avec le degré d'absorption. Dans la forme la plus simple, l'intensité lumineuse $I(r)$ varie en fonction de la loi du carré inverse $I(r) = \frac{I_s}{r^2}$ où I_s est l'intensité à la source. Pour un milieu donné avec un coefficient fixe d'absorption g, l'intensité lumineuse I varie avec la distance r où :

$$I = I_0 e^{-\gamma r}$$

Où I_0 est l'intensité de la lumière d'origine.

Comme l'attractivité d'une luciole est proportionnelle à l'intensité de la lumière vue par des lucioles adjacentes, l'attractivité β d'une luciole peut être définie par:

$$\beta = \beta_0 e^{-\gamma r^2}$$

Où β_0 est l'attractivité pour $r = 0$.

Pseudo-code du FA basique

Begin
  Fonction objectif $f(x), x = (x_1, \dots, x_d)^T$
  Générer une population initiale de lucioles $x_i (i = 1,2, \dots, n)$
  L'intensité de la lumière I_i à x_i est détermine par $f(x_i)$
  Définir le coefficient d'absorption de la lumière γ
  while ($t <MaxGeneration$)
    for $i = 1{:}\, n$ les n lucioles do
      for $j = 1{:}\, i$ les n lucioles do
        if ($I_j > I_i$) then
          Faire déplacer la luciole i vers j dans les d-dimensions
        end
        L'attractivité varie avec la distance r via $exp[-\gamma r]$
        Evaluer les nouvelles solutions et mettre à jour l'intensité de la lumière
      end
    end

Classer les lucioles et trouver la meilleure
End
End

3.2.3.5 Systèmes immunitaires artificiels AIS

Les systèmes immunitaires sont des techniques de calcul inspirées par le système immunitaire biologique qui peuvent être utilisées pour résoudre des problèmes complexes du monde réel. La technique de l'AIS fait évoluer des solutions améliorées du problème par le moyen de la sélection clonale, la théorie des réseaux immunitaires, la vaccination ou d'autres concepts du système immunitaire.

En général, un algorithme d'optimisation immunitaire doit avoir une population d'individus (solutions candidates) appelées anticorps et d'autres individus (ou objectifs) que les anticorps tentent de les atteindre ou de les correspondre qui sont appelés antigènes. Les principales différences entre les techniques de l'AIS appliquées aux problèmes d'optimisation résident dans la façon dont leurs anticorps évoluent.

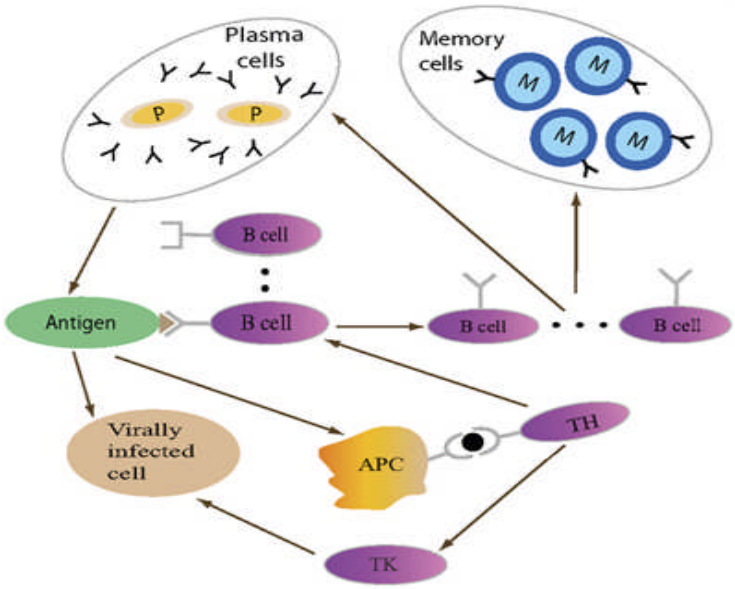

Figure 3.12 : Les Lymphocytes (comme les cellules B et T) reconnaissent et éliminent les antigènes dans un système immunitaire naturel.

L'algorithme CLONALG [De Castro 2002] fait évoluer les anticorps inspirés par le concept de la sélection clonale. L'évolution de la sélection clonale est basée sur le principe que chaque individu est cloné et hyper-muté et ceux qui ont une plus grande affinité sont sélectionnés. Le taux de mutation est normalement inversement proportionnel à l'affinité de

l'anticorps par rapport aux antigènes. Généralement les AIS n'utilisent pas les opérateurs de recombinaison (comme le croisement dans les GA).

L'algorithme opt-aiNET Basé sur la théorie des réseaux immunitaires est une autre technique immunitaire d'inspiration bien connue. L'idée principale de cette méthode est de construire un réseau des meilleurs anticorps non similaires à savoir maintenir dans le système immunitaire des cellules du corps (soi) qui ont moins d'affinité entre eux et plus d'affinité avec celles produites par des intrus ou par les cellules sous le contrôle du virus (non-soi).

Keko et al [ICeko 2003] ont eu de bonnes solutions à des problèmes d'optimisation combinatoire en incluant un processus de vaccination dans leur algorithme AIS qui peut être considéré comme un moyen d'introduire la connaissance du domaine dans l'algorithme.

3.2.3.6 Etat de l'art sur les approches dynamiques pour l'intelligence distribuée

Après la détection des changements dans l'environnement, il doit y avoir une stratégie pour répondre efficacement à une grande variété de changements. Pour cela, Eberhart et Xiahui Hu [Eberhart 2001] ont utilisé une méthode basée sur la réinitialisation de la mémoire de la population dans leur algorithme basé PSO. En 2002 ils ont constaté que cette méthode peut poser quelques problèmes, par exemple : toute la population converge vers une petite zone de l'espace de recherche. Alors ils ont proposé une version PSO adaptative qui consiste à utiliser une diversification aléatoire d'une partie de la population pour traquer automatiquement plusieurs changements dans un système dynamique.

Stefan Janson et Martin Middendorf [Janson 2006] présentent un mécanisme qui maintient une hiérarchie de particules appelé *PSO hiérarchique partitionné* (PH-PSO). Après qu'un changement de l'environnement survienne, l'essaim est partitionné en plusieurs sous-essaims pour un certain nombre de générations. Le but de cette méthode est d'éviter que l'essaim converge vers l'ancienne position de l'optimum global très rapidement.

Plus récemment, Moayed Daneshyari et Gary G. Yen [Daneshyari 2011] ont proposé un algorithme basé-culture en utilisant les connaissances stockées dans un espace de croyance pour diversifier et repousser la population juste après la détection du changement. Ainsi, l'algorithme peut facilement calculer le facteur de répulsion pour chaque particule et de localiser les particules dirigeantes dans les trois niveaux: le niveau personnel, le niveau essaim et le niveau global. Chaque particule du PSO dynamique basée-culture volera à travers un mécanisme de ces trois niveaux de vol incorporés avec un facteur de répulsion.

Après qu'un changement survienne, les particules se regroupent en plusieurs essaims et une migration fondée sur la diversité entre les essaims avec un mécanisme de répulsion sera mis en place pour accroître la diversité aussi rapidement que possible. Une étude comparative à travers les résultats expérimentaux montre que le nouvel algorithme surpasse plusieurs versions dynamiques du PSO.

Une technique qui utilise une répulsion inter-particules (évitement des collisions) appelée Charged PSO (CPSO) est proposée par Blackwell et Bentley [Blackwell 2002]. Un essaim est constitué de particules "*chargées*" et d'autres "*neutres*". Les particules chargées se repoussent et forment un nuage de particules chargées autour du noyau, cette charge améliore la diversité dans le voisinage du sous essaim convergeant.

Une idée similaire du CPSO a été utilisée par Branke et Blackwell [Branke 2004] dans *Quantum PSO* (QPSO). Les particules quantiques de chaque sous essaim ont été introduites comme un moyen de maintenir un certain niveau de diversité au sein de l'essaim, elles ont été inspirées par des modèles atomiques. Dans l'image de l'atome quantique proposée, les électrons ne circulent pas dans des orbites déterministes mais ils sont distribués en nuage (une probabilité autour du noyau). L'atome du PSO se compose d'un noyau de particules exécutant un PSO normal selon les règles normales de mise à jour, typiquement ce noyau sera rétrécit dans la taille car il converge vers un optimum. Le noyau est entouré par des particules quantiques, ces particules quantiques ne suivent pas les règles du PSO mais sont placées dans des positions autour du centre du noyau selon une distribution de probabilité. En conséquence, elles ne convergent pas mais maintiennent un niveau constant de la diversité.

Dans le modèle *Compound PSO* (CPSO) proposé par Lili Liu et al [Liu 2008], un certain nombre de particules composées sont créées. Chaque particule composée est créée en tant que simple structure géométrique qui se compose de trois particules: une particule est aléatoirement choisie dans l'essaim initial et les deux autres sont générées d'une façon aléatoire pour former un triangle. Les trois particules dans une particule composée sont désignées comme "*des particules membres*". Les particules dans l'essaim qui n'appartiennent à aucune particule composée sont désignées comme "*particules indépendantes*". Chaque particule composée ajuste sa structure interne afin de suivre la trace de l'optimum qui change. Les étapes essentielles concernent la construction d'une nouvelle particule composée pour explorer les bonnes solutions dans un nouvel environnement et identifier une "*particule*

représentative" pour chaque particule composée afin de participer au PSO canonique. Afin de maintenir la diversité ainsi de garantir la précision de la recherche dans les particules composées, deux facteurs sont intégrés pour identifier la particule représentante: le premier est le *fitness* et l'autre est la *distance* totale entre une particule membre par rapport aux deux autres particules. Dans une étude expérimentale CPSO surpasse SPSO et PSO dans la plupart des tests dynamiques effectués.

Hongfeng Wang et al [Hongfeng 2007] ont utilisé l'idée du modèle de la mémoire Tri-Island. Dans leur méthode "*Triggered Memory-Based PSO*" basée-mémoire, l'algorithme procède par réinitialiser la *Population-Recherche* et récupérer la mémoire chaque fois qu'un changement environnemental est détecté. Cependant, il peut y avoir quelques problèmes pour ce schéma. Lorsque l'environnement change lentement, la population aurait toujours séjournée dans un pic au lieu de chercher d'autres sommets pour une longue durée. Pour résoudre ce problème, ils ont proposé un nouveau système de "*mémoire-déclenchée*" pour le PSO basé-mémoire dans des environnements dynamiques où chaque fois que la *Population-Recherche* trouve un pic, elle sera immédiatement réinitialisée et la mémoire sera récupérée.

3.2.3.3 Conclusion

Une tendance prometteuse consiste à adopter des techniques qui ont fait leurs preuves pour les problèmes d'optimisation dynamique. En particulier, les techniques basées sur l'adaptation des paramètres et les techniques à plusieurs populations semblent être les plus prometteuses et les plus utilisées. Selon l'examen de la littérature, nous pouvons conclure que les différentes approches semblent être meilleures pour différents types de problèmes. C'est la raison pour laquelle de nombreuses études récentes tentent de combiner plusieurs méthodes dans un seul algorithme pour mieux résoudre les problèmes dynamiques.

Chapitre 4

Nouvelles approches pour l'optimisation dynamique basées BFO

4.1 Introduction

Beaucoup de problèmes du monde réel sont dynamiques, ce qui nécessite des algorithmes d'optimisation qui sont capable de suivre en permanence un optimum qui change au fil du temps. Ces algorithmes doivent être adaptés pour avoir des résultats optimaux sur des problèmes d'optimisation dynamique. Si la population est convergente, les attracteurs seront proches de la position optimale et la population se rétrécit à un rythme déterminé par le facteur de constriction et par l'environnement local de l'optimum. Si le décalage de l'optimum dans la population s'effondre alors la ré-optimisation sera efficace. Toutefois, si le décalage de l'optimal est significativement loin de la population, la poursuite va être inhibée et la population peut même osciller hésiter entre un faux attracteur et le véritable optimum.

Le compromis entre la convergence rapide et être piégé dans des optima locaux sera encore plus critique dans les fonctions multimodales qui ont de nombreux optima locaux très proches les uns des autres. Afin d'échapper aux optima locaux et éviter la convergence prématurée, la recherche d'un optimum global devrait être diversifiée. De nombreux chercheurs ont amélioré la performance des différentes approches en améliorant leur capacité avec une recherche plus diversifiée à travers l'utilisation de plusieurs populations.

Dans ce chapitre nous présentons deux nouvelles approches, la première approche dynamique que nous allons présenter est basée sur l'algorithme BFO; cette version dynamique (DBFO) de cet algorithme est étendue à une version auto-adaptative nommée (DABFO) en introduisant des techniques adaptatives de prédiction et de maintien de la diversité. Ces techniques sont basées sur la trajectoire, la vitesse et l'évolution de la valeur du meilleur fitness. Ces deux versions d'algorithmes sont appliquées pour traiter le problème de la poursuite d'objets en utilisant la similarité des histogrammes. La deuxième approche (MBFO) est une approche multi-population basée BFO. Cette approche est évaluée avec des scénarios du benchmark multimodal et dynamique *Moving Peaks Benchmark* (*MPB*).

4.2 BFO dynamique auto-adaptative

Tout d'abord, un BFO dynamique (DBFO) similaire à [Nguyen 2012] est présenté. Quelques modifications sont introduites au BFO original par une réévaluation du coût des bactéries au début de chaque étape de « dispersion et élimination » et par une réévaluation de la meilleure position déjà enregistrée dans la fin de l'étape de "*dispersion et élimination*" (la valeur de la fonction objectif dans une même position peut être modifiée au cours du temps dans un environnement dynamique). Nous allons ensuite étendre cette version dynamique à une version auto-adaptative nommée DABFO qui emploie un paramètre de *dispersion-élimination* auto-adaptatif qui garantit un bon maintien de l'équilibre de la diversité et une exploration / exploitation efficace de l'espace de recherche pour faire face aux changements de l'environnement dynamique. DABFO utilise également une technique de prédiction auto-adaptative qui sera présentée dans les paragraphes suivants.

4.2.1 Elimination-dispersion auto-adaptative

Afin de maintenir une diversité flexible des bactéries dans l'espace de recherche, le paramètre de probabilité fixe de l'élimination et de la dispersion P_{ed} est remplacée par un autre qui est dynamique, sa valeur est déterminée à partir du fitness de la meilleure solution courante et à partir des limites de la meilleure solution dans les états précédents (f_{c_max} et f_{c_min}), ce paramètre de probabilité de l'élimination et de la dispersion est calculé pour être entre la probabilité maximale et minimale ($P_{ed_{max}}$ et $P_{ed_{min}}$) dans l'équation suivante:

$$P_{ed} = Coef \times \left(P_{ed\_max} - P_{ed\_min}\right) + P_{ed\_min}$$

$$Coef = \frac{\left(f(Cell_{best}) - f_{min}\right)}{\left(f_{c\_max} - f_{c\_min}\right)}$$

Où P_{ed_max} et P_{ed_min} sont la probabilité maximale et minimale d'une cellule étant soumise à l'élimination et à la dispersion.

Ce paramètre auto-adaptatif P_{ed} joue un rôle important pour contrôler la proportion des individus destinés à la diversité; lorsque l'optimum est poursuivi de près (bon fitness) P_{ed} se fait automatiquement réduit pour favoriser l'exploitation par la reproduction et la Chimiotaxie, lorsque l'optimum est plus loin (mauvaise qualité du fitness) P_{ed} est augmenté pour permettre plus de diversité et de découvrir d'autres régions dans l'espace de recherche.

4.2.2 Prédiction auto-adaptive

Cette technique assure une poursuite rapide et efficace basée sur trois éléments auto-adaptatifs:

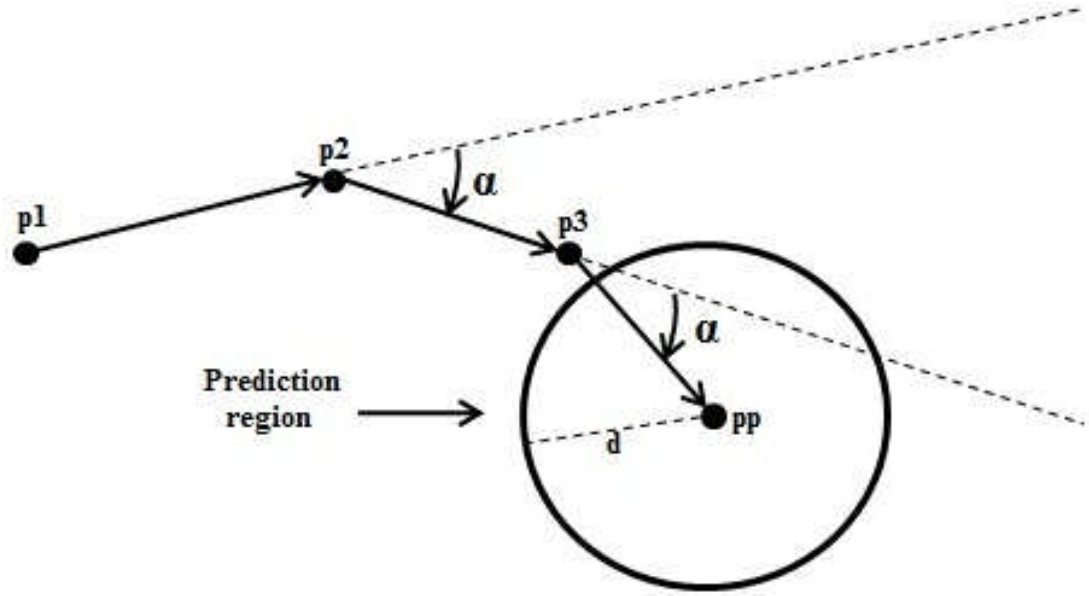

Figure 4.1 : Prédiction auto-adaptative.

4.2.2.1 La position de la région de prédiction

Basée sur la trajectoire et le changement de la vitesse de l'optimum poursuivi, cette position p_p sera déterminée à partir des trois dernières positions optimales mémorisées

auparavant p_1, p_2 et p_3. Un angle α doit être calculé, ensuite la taille d_0 du vecteur $\overrightarrow{p_2p_3}$ est utilisée avec α pour former un nouveau vecteur $\overrightarrow{p_3p_p}$ qui prédit la nouvelle position p_p comme il est illustré dans la figure 4.1, dans laquelle p_p représente la position où la partie de la population destinée pour la prédiction sera placée de manière aléatoire autour de cette nouvelle position.

Où $p_{p.x}$ et $p_{p.y}$ sont définis par les équations suivantes:

$$p_{p.x} = d_0 \times \cos(\beta) + p_{3.x}$$

$$p_{p.y} = d_0 \times \sin(\beta) + p_{3.y}$$

$$\beta = 2 * \operatorname{atan}(p_{3.y} - p_{2.y}, p_{3.x} - p_{2.x}) - \operatorname{atan}(p_{2.y} - p_{1.y}, p_{2.x} - p_{1.x}),$$

$$d_0 = \sqrt{(p_{3.x} - p_{2.x})^2 + (p_{3.y} - p_{2.y})^2},$$

4.2.2.2 La taille de la région de prédiction

La région de prédiction est définie par un cercle de centre p_p et un diamètre d qui est variable (défini à partir de la dernière vitesse observée de l'optimum). Par conséquent, la taille de la région de prédiction est automatiquement adaptée à la vitesse optimale en utilisant l'équation :

$$d = C_p \times d_0$$

Où C_p est un coefficient constant.

4.2.2.3 Le nombre d'individus destinés à la prédiction

Ce nombre peut être déterminé par la même idée proposée dans notre chapitre 4.2.1 et pour les mêmes raisons, il est calculé comme suit :

$$p_{nbr} = Coef \times (P_{max} - P_{min}) + P_{min}$$

Où P_{max} et P_{min} sont la proportion maximale et la proportion minimale de la taille de la population destinée pour la prédiction.

- **Pseudo code de l'algorithme DABFO**

Population ← InitializePopulation($Cells_{num}$, $Problem_{size}$);
for l = 0 to N_{ed} do
  Reevaluate(Population);
  for k = 0 to N_{re} do
    for j = 0 to N_c do
      ChemotaxisAndSwim(Population, $Problem_{size}$, $Cells_{num}$, N_s, $Step_{size}$, $d_{attract}$, $w_{attract}$, $h_{repellant}$, $w_{repellant}$);
      foreach *Cell* ∈ Population do
        if Cost(*Cell*) ≤ Cost($Cell_{best}$) then
          $Cell_{best}$ ← *Cell*;
        end
      end
    end
    SortByCellHealth(Population);
    Selected ← SelectByCellHealth(Population, $Cells_{num}$);
    Population ← Selected+ Selected;
  end
  Reevaluate($Cell_{best}$);
  f_{max}, f_{min} ← CalculateCostMaxMin($Cell_{best}$);
  $Coef$ ←CalculateCoefficient(f_{max}, f_{min}, Cost($Cell_{best}$));
  P_{ed} ← AdaptEliminationDispersal($Coef$, P_{ed_max}, P_{ed_min});
  foreach *Cell* ∈ Population do
    if Rand() ≤ P_{ed} then
      Cell ← CreateCellAtRandomLocation();
    end
  end
  p_1, p_2, p_3 ← SheeftAndMemorizeLastBest($p_1, p_2, p_3, Cell_{best}$);
  p_p, d ← Prediction(p_1, p_2, p_3);
  p_{nbr} ← PredictionPartNumber($Coef$, P_{max}, P_{min});
  foreach *Cell* ∈ at low p_{nbr} of Population cells do
    Cell ← CreateCellAtPredictionRegion(p_p, d);
  end
end
return $Cell_{best}$;

4.2.3 Application du DBFO et DABFO pour le problème de la poursuite d'objets

La poursuite d'objets peut être considérée comme un problème d'optimisation dynamique (DOP) à la fois à cause du changement au fil du temps de l'environnement mais aussi de la position de l'objet cible qui change.

Alors que certaines caractéristiques de l'objet, comme la forme, la rotation ... etc, changent sévèrement dans un environnement bruité. Utiliser la méthode basée sur l'espace de couleur est un bon choix en la comparant à d'autres méthodes telles que la poursuite basée région, la poursuite basée contour actif et la poursuite basée modèle. Dans la méthode de la poursuite basée espace de couleurs, les algorithmes utilisent les informations disponibles dans l'espace de couleurs pour établir des métriques à des fins de poursuite comme il est le cas de la technique que nous avons choisi (similarité des histogrammes de couleurs) où les histogrammes changent légèrement avec le changement d'angle de vue et d'échelle. Cette technique utilise une région rectangulaire sélectionnée comme référence pour la poursuite. L'objet cible peut être poursuivi par une corrélation de correspondance entre la fenêtre de la région référence de l'image et une fenêtre secondaire de la région candidate de l'image, où une quantification de la similarité des deux groupes de valeurs de pixels est calculée et comparée.

4.2.3.1 Histogramme de couleur

La couleur est largement utilisée pour décrire l'objet en raison de son invariance de la translation, de la rotation et de l'échelle. Les espaces de couleur YCbCr et HSV ont des performances de poursuite systématiquement plus élevées par rapport à l'espace de couleur du niveau de gris; les travaux sur le problème de la poursuite montrent que l'utilisation d'une seule couche, où l'information de couleur a été stockée pourrait être utilisée pour le problème de la poursuite d'objets. L'information de la couleur dans l'espace couleur YCbCr est stockée dans les couches Cb et Cr tandis que dans l'espace couleur HSV elle est stockée dans les couches de la teinte (H) et de la saturation (S), la teinte d'une couleur se réfère à la longueur d'onde spectrale qui correspond étroitement aux couleurs de l'arc en ciel. Typiquement une teinte de 0 est rouge, qui est également utilisée comme valeur de référence, 120 est vert, 240 est bleu. Le paramètre de saturation (S) est la mesure de la pureté de la couleur, le paramètre de valeur (V) indique les niveaux de luminosité ou de la plage des valeurs de gris du noir au blanc.

Contrairement au RGB, HSV sépare Luma (l'intensité de l'image) du Chroma (les informations de la couleur), ce qui est très utile dans la poursuite basée-couleur. Dans notre travail, nous utilisons la similarité des histogrammes de la couche Hue de l'espace de couleur HSV.

Un histogramme d'une image est un type d'histogramme qui agit comme une représentation graphique de la distribution tonale d'une image numérique. Il trace le nombre de pixels pour chaque valeur tonale. Chaque bactérie représente la position d'une région de fenêtre candidate. Initialement une région de référence de la fenêtre rectangulaire sélectionnée est utilisée pour la poursuite au sein d'une image de la vidéo. Ensuite, l'objet cible peut être atteint par une correspondance de corrélation entre les histogrammes de couleur du niveau teinte (Hue) de la fenêtre de référence sur l'image et la sous-fenêtre de la région candidate de l'image en rassemblant les valeurs de teinte H des pixels de l'image cibles comme il est montré dans les figures 4.2 et 4.3.

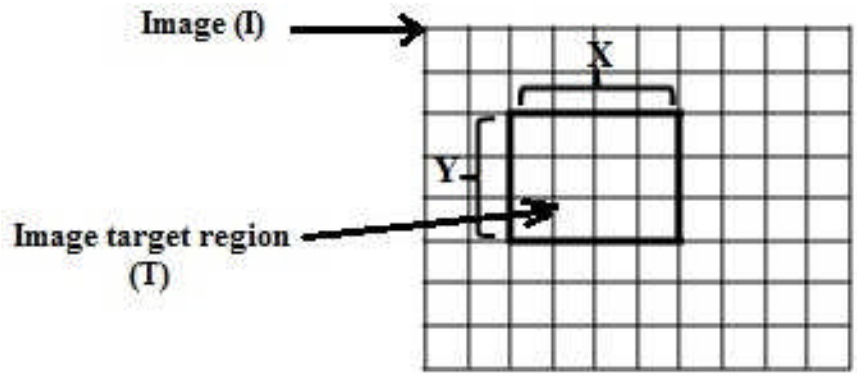

Figure 4.2 : Exemple d'une fenêtre de la sous région candidate d'une image de la vidéo.

Où X représente la largeur en pixels de T, Y représente la hauteur en pixels de T, s est le nombre d'intervalles du niveau de la teinte dans l'histogramme, Hi est le nombre de pixels de la région cible de l'image (T) qui figure dans le i^{ème} intervalle, Hi peut prendre des valeurs entre $[0, X * Y]$, de l'autre côté, il est clair que $\sum H_i$ est égal à $X * Y$.

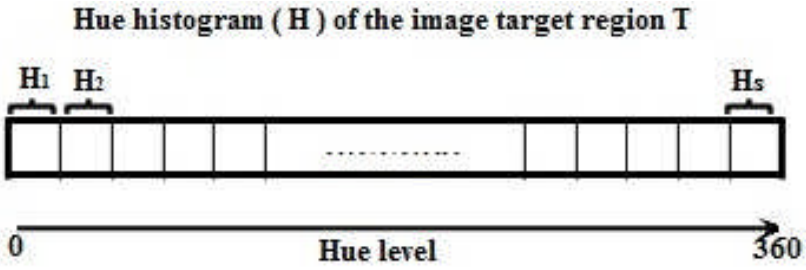

Figure 4.3 : Histogramme du niveau teinte.

Pour l'identification d'un objet dans une trame, une fenêtre rectangulaire est utilisée pour représenter la position d'une bactérie candidate. Dans chaque itération de la procédure élimination-dispersion, la bactérie qui possède le meilleur fitness détermine l'objet cible de la trame courante. Chaque image dans une vidéo est traitée continuellement par l'algorithme sans réinitialiser les paramètres.

4.2.3.2 Formulation de la fonction objectif

La fonction objectif f utilisée dans le présent travail est calculée comme suit (fonction de minimisation) : Initialement, un histogramme de couleur du niveau teinte $H_{Ref}(T_0)$ de la région de l'objet sélectionné à poursuivre est extrait, puis il sera stocké pour être utilisé comme une référence, où T_0 est l'image référence de la fenêtre sélectionnée. Pour calculer le fitness d'une région d'image candidate Ti au cours du processus d'optimisation, un autre histogramme (H_i) devrait être calculé pour elle. La similarité entre les deux histogrammes H_{Ref} et H_i est calculée comme la valeur absolue de la différence des deux histogrammes. Cette similarité aura une valeur de 0 à (2.X.Y), 0 correspond à une totale similarité et (2.X.Y) à aucune similarité. La fonction objectif f est normalisée à une valeur comprise entre 0 et f_{max} (dans nos algorithmes f_{max}= 50) comme suit :

$$f(T_i) = \frac{f_{max} \times \sum_{j=1}^{s} \text{abs}\big(H_i(j) - H_{Ref}(j)\big)}{2 \times T_{i.x} \times T_{i.y}}$$

4.2.4 Expériences et résultats

4.2.4.1 Expériences et configuration de l'environnement

Les expériences sont réalisées dans un environnement de test de 3 vidéos (A, B et C) du Benchmark publique BoBoT pour les algorithmes et les systèmes de poursuite (disponibles dans [Klein 2013], qui comprend plusieurs courtes séquences de vidéos montrant des objets cibles arbitraires en mouvement.

Les expériences ont été utilisées pour toutes les trames de chaque vidéo. Figure 4.5, 4.6 et 4.7 montrent seulement les 120 premières images de chaque vidéo. Toutes les vidéos ont un format de 320x240 et 25fps. Les algorithmes ont été implémentés en MATLAB R2012a.

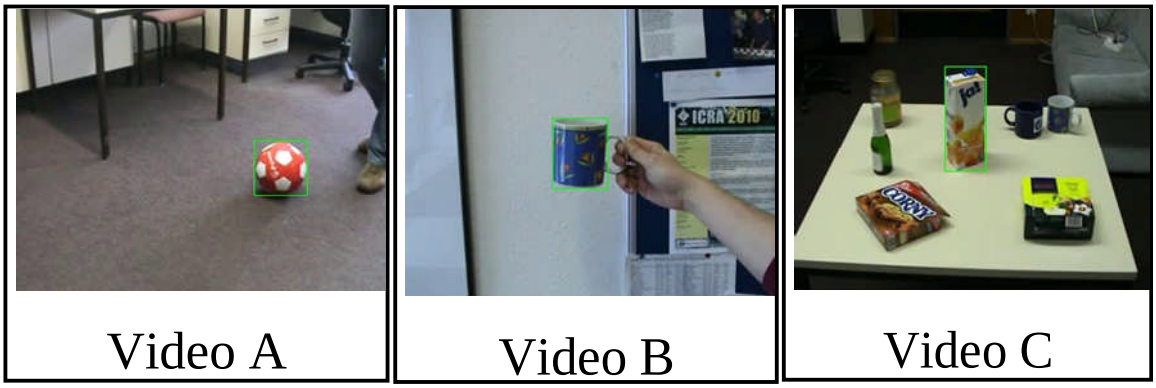

Figure 4.4 : Exemple de trames de 3 vidéos du benchmark BoBoT [Klein 2013].

4.2.4.2 Métriques de performances utilisées

Les mesures de la performance de précision (Accuracy) et le fitness moyen collectif FC sont utilisées (dans notre cas f_{min}, f_{max} sont connues : 0 et 50).

$$Accuracy_i = f\left(generation_{best_i}\right) - f_{min}$$

$$F_C = \frac{1}{N} \times \sum_{i=1}^{N} f\left(generation_{best_i}\right)$$

Dans toutes les expériences les paramètres suivants sont initialisés comme suit :

$Cells_{num} = 20,\ N_{ed} = 1,\ N_{re} = 1,\ N_c = 4,\ N_s = 3,\ P_{ed} = 0.25,\ P_{ed\_max} = 0.4$,$P_{ed\_min} = 0.1,\ P_{max} = 0.6,\ P_{min} = 0.2$,$C_p = 1.5$.

4.2.4.3 Résultats expérimentaux

Tous les tests sont obtenus d'une moyenne de 30 exécutions. Vidéo A, B et C ont respectivement les tailles de 602, 629 et 404 trames. Les figures 4.5 , 4.6 et 4.7 montrent les graphes de performance moyenne de la précision des algorithmes DBFO et DABFO dans les 120 premières trames de chaque vidéo. Le fitness moyen collectif dans chaque vidéo est récapitulé dans le tableau 4.1, Le tableau 4.2 représente des statistiques comparatives sur le nombre de trames mieux poursuivies par DABFO que celles poursuivies par DBFO dans chaque vidéo.

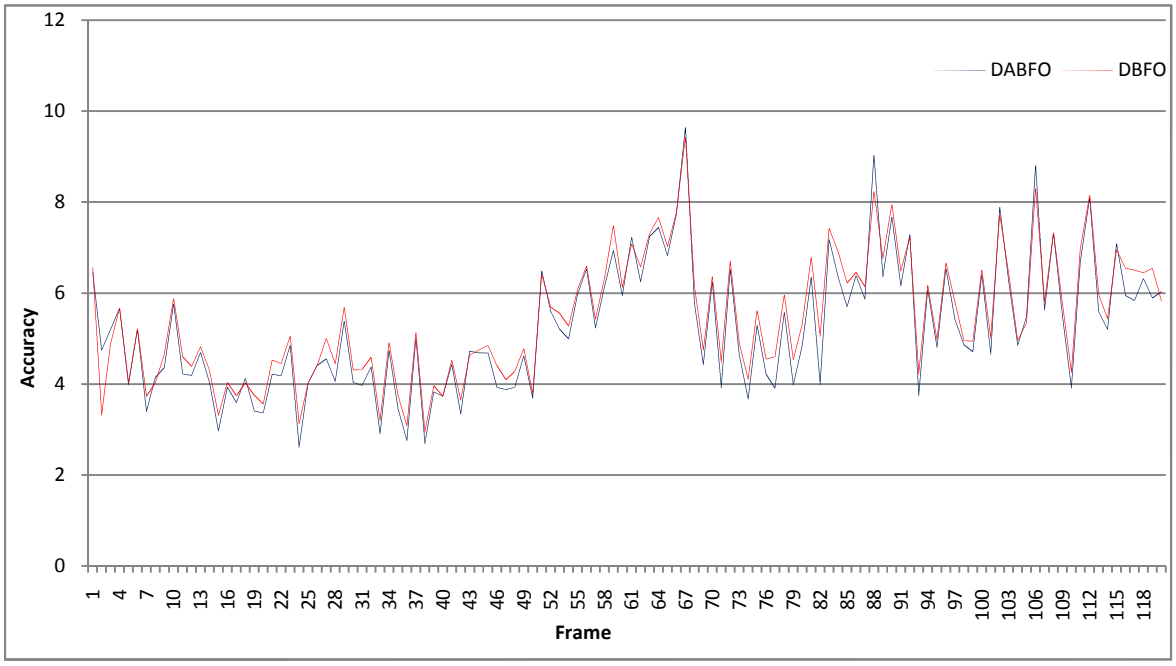

Figure 4.5 : Fitness moyen collectif de la vidéo A.

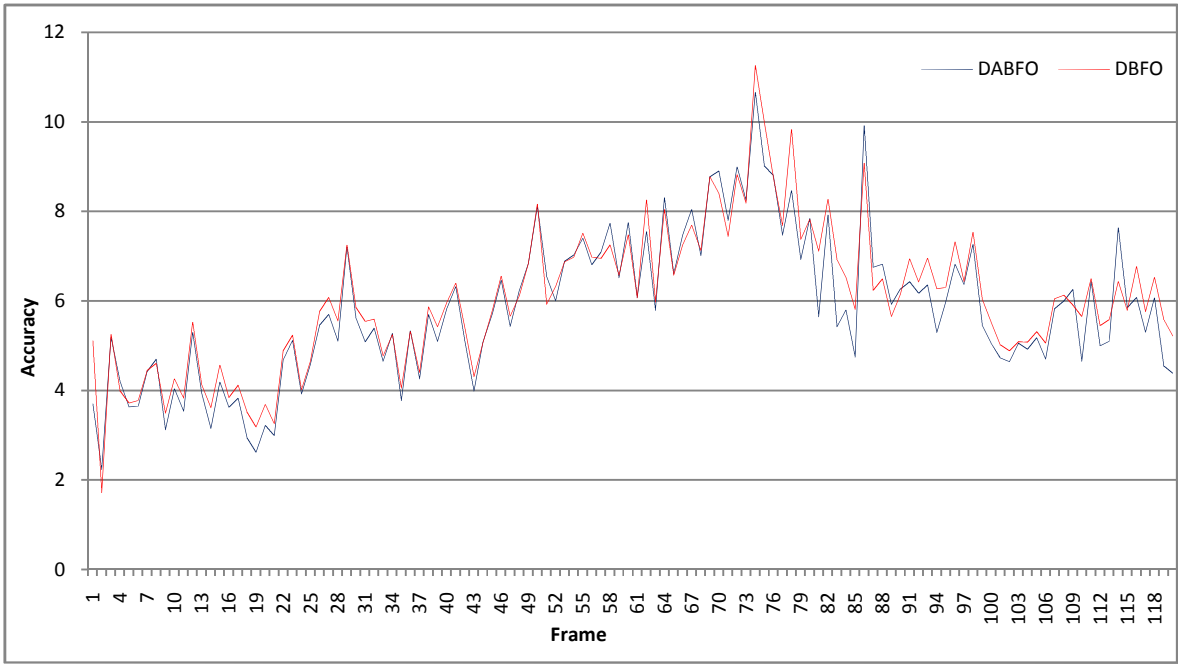

Figure 4.6 : Fitness moyen collectif de la vidéo B.

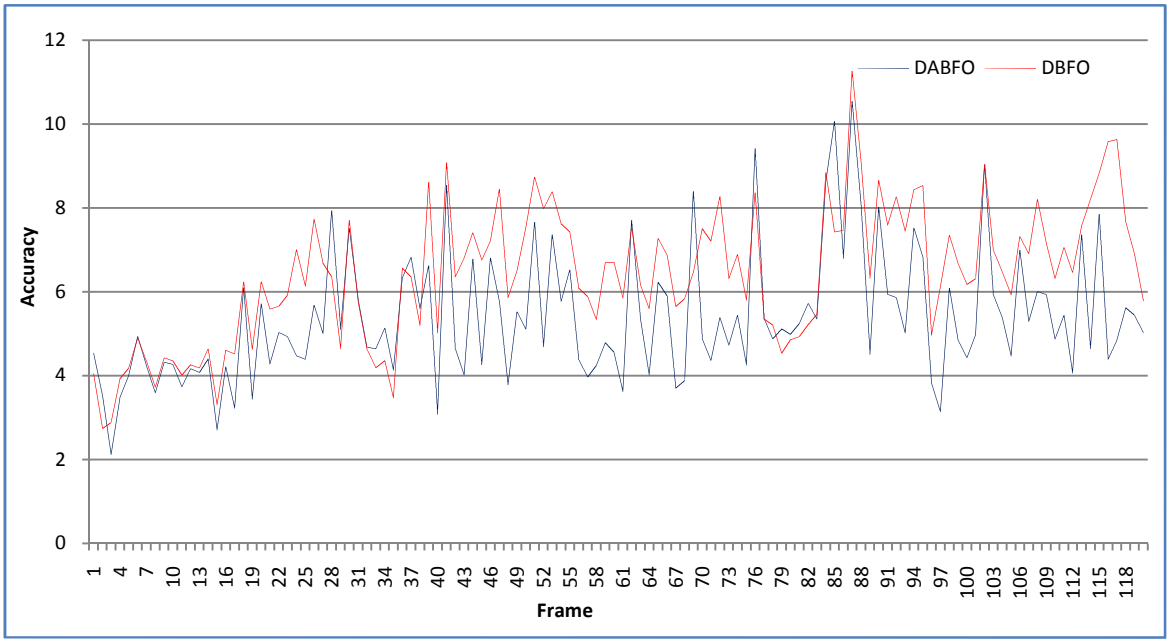

Figure 4.7 : Fitness moyen collectif de la vidéo C.

Algorithme	**Moyenne du fitness collectif moyen**		
	Video A	***Video B***	***Video C***
DBFO	6.0350	10.1135	8.0972
DABFO	5.7932	9.8888	7.5305

Table 4.1 : Fitness moyen collectif.

Nombre de trames mieux suivies par DABFO que par DBFO (Nbre de trames/ Nbre de trames total)		
Video A	***Video B***	***Video C***
527/602	500/629	291/404

Table 4.2 : Statistiques sur le nombre de trames mieux poursuivies.

4.2.5 Discussion et analyse des résultats

Dans toutes les expériences, les deux algorithmes DBFO et DABFO peuvent poursuivre l'objet dans toutes les trames de chaque vidéo avec une bonne précision où l'objet est poursuivi continuellement. Malgré que les deux algorithmes DBFO et DABFO utilisent les mêmes valeurs de paramètres et le même nombre d'itérations du processus d'exécution, la précision de la poursuite de DABFO surpasse nettement DBFO comme l'indique le tableau

4.1. En outre, lorsque la gravité et la vitesse du changement sont grandes et pas stables, cas de la vidéo C par rapport aux vidéos A et B, les résultats montrent que DABFO surpasse DBFO significativement comme il est illustré dans la figure 4.7, où DABFO s'auto-adapte rapidement aux changements de l'environnement et montre plus de performance. Le tableau 4.2 montre qu'environ 80% des trames de vidéos sont mieux poursuivies en utilisant l'algorithme DABFO.

Par ailleurs, nous pouvons appliquer notre approche dans des domaines variés allant de l'imagerie médicale aux applications militaires et autres. En effet, nous avons testé notre approche sur la poursuite des missiles et avions dans le domaine militaire, aussi bien dans le domaine biologique dont l'intérêt est d'étudier le comportement de certaines cellules microscopiques (telles que les bactéries et les virus) et encore plus dans plusieurs domaines de la vie quotidienne et éducative comme le cas du suivi du doigt pour l'écriture dans les tableaux magiques.

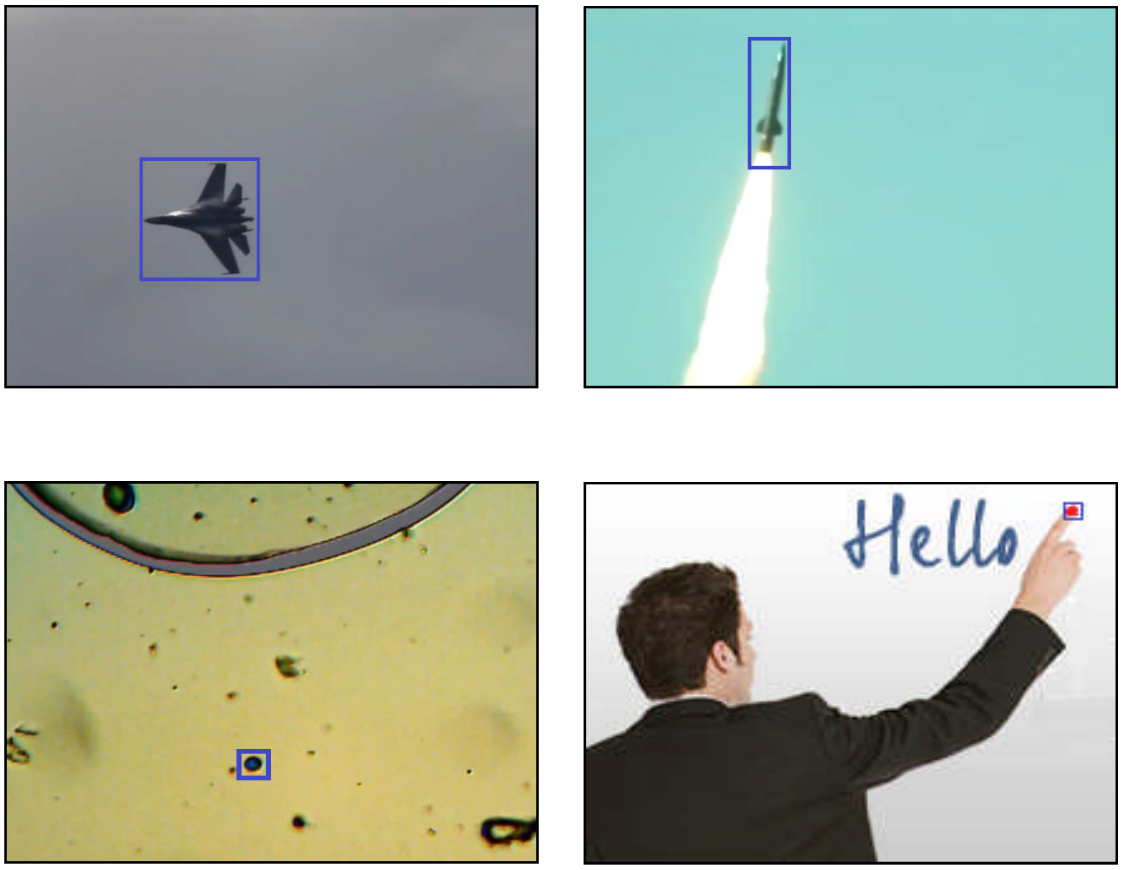

Figure 4.8 : Quelques applications du DABFO.

4.2.6 Conclusion

Dans ce travail, nous avons présenté une version dynamique de l'algorithme BFO pour le problème de la poursuite d'objets en utilisant la méthode de similarité des histogrammes du niveau teinte (H) de l'espace de couleur HSV. Aussi une version auto-adaptative du BFO dynamique appelée DABFO est présentée, dans laquelle plusieurs techniques auto-adaptatives ont été introduites (la position de la région de prédiction, la taille de la région de prédiction, le nombre d'individus destinés à la prédiction et le paramètre de probabilité d'élimination et de dispersion auto-adaptative) afin d'améliorer le compromis exploration-exploitation et de s'adapter aux changements de l'environnement. Notre algorithme DABFO rivalise et surpasse même les algorithmes existants basés BFO surtout quand la gravité et la vitesse des changements sont grandes.

4.3 BFO multi-population proposé (MBFO)

L'algorithme BFO est une sorte d'algorithmes d'intelligence distribuée. Cet algorithme a été amélioré pour optimiser certains problèmes statiques, une description détaillée de cette algorithme est présentée dans le chapitre 3.

L'idée principale de la technique de Multi-population est de diviser la population en un nombre de sous populations, dans le but de poser chacune de ces sous populations sur des sommets prometteurs différents dans le paysage de la fonction objectif. Si une population est divisée en un certain nombre de sous populations, il peut arriver que des particules de différentes populations se regroupent autour d'un seul pic. Cela n'est pas souhaitable car la motivation derrière une approche Multi-population est de poser les différentes populations sur les différents sommets, pour cela un mécanisme d'exclusion est introduit. BFO présente un bon mécanisme de maintien de la diversité, afin de l'adapter aux environnements dynamiques nous modifions l'algorithme par déclencher l'étape de l'élimination et dispersion à la détection d'un changement au lieu de le faire d'une manière itérative.

4.3.1 Exclusion dans MBFO

L'exclusion est une interaction entre les différentes sous populations pour prévenir ces différentes sous-populations d'exploiter les mêmes régions, dans notre BFO multi-population un rayon d'exclusion R (prédéfinit) est vérifié pour garantir que deux sous-populations n'exploitent pas la même région, à chaque étape d'élimination et de dispersion. Si la distance D_{ij} entre la position dont la qualité des deux meilleurs individus des deux populations i et j est inférieure au rayon d'exclusion R, la population dont la qualité du meilleur individu est inférieure est réinitialisée aléatoirement. La distance entre deux sous populations est calculée comme suit :

$$D_{ij} = \sqrt{\sum_{k=1}^{dim} (P_{ik} - P_{jk})^2}$$

Où : dim est la dimension du problème.

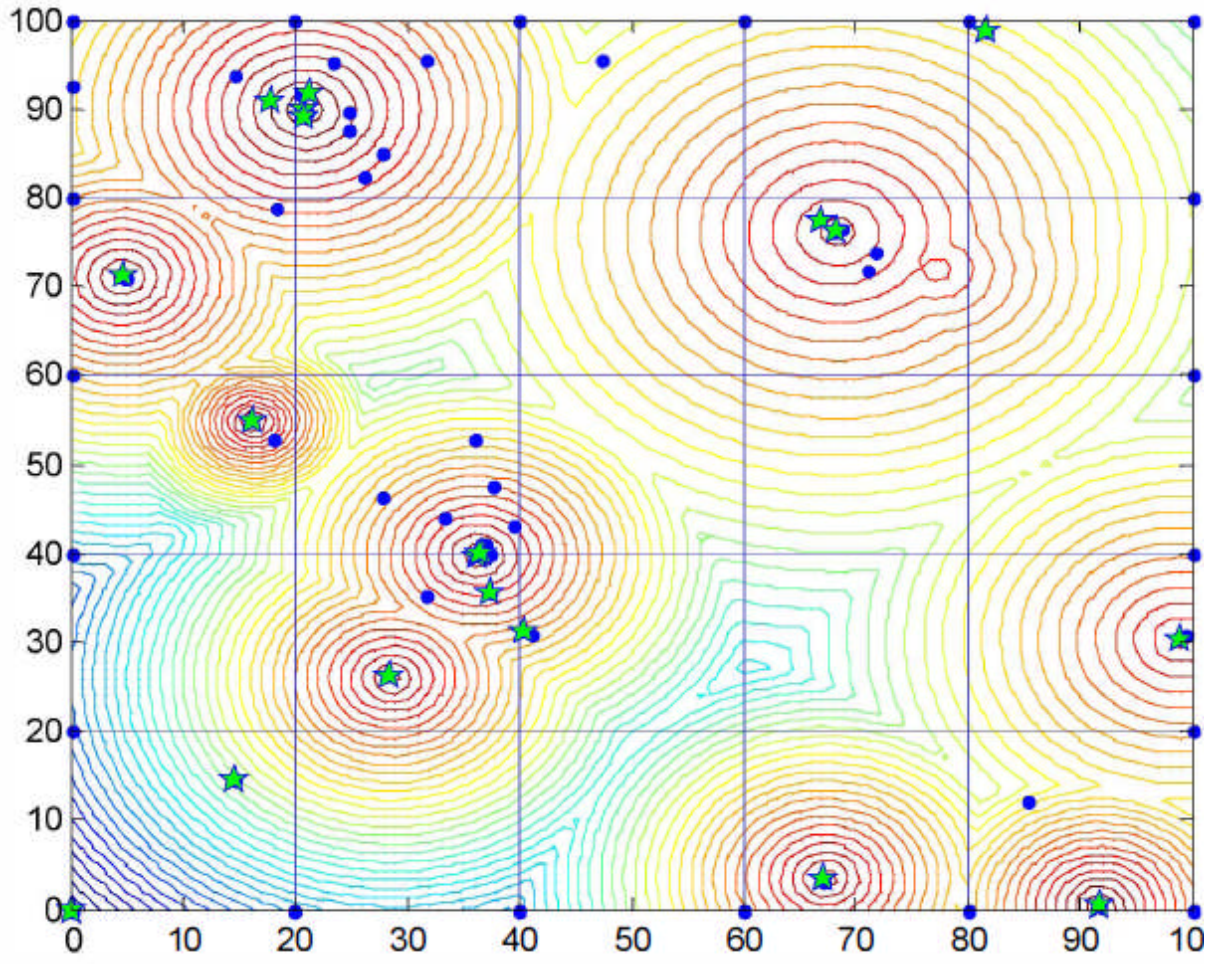

Figure 4.9 : Exemple de l'exclusion dans le model multi-population [Nabizadeh 2012].

4.3.2 Réactivité aux changements et maintien de la diversité

En addition du mécanisme de l'élimination et dispersion qui caractérise BFO pour garantir une bonne diversité, la notion de multi-population assure non seulement une poursuite multimodal mais elle permet aussi de renforcer l'aspect de maintien de la diversité en couvrant mieux l'espace de recherche durant le processus d'optimisation en utilisant l'exclusion entre les sous populations et la notion de choisir la mauvaise population pour être réinitialisée. Pour une bonne exploitation-exploration, le système répond aux changements par une réaction de déclenchement d'une réévaluation du fitness des individus de la population. Pour détecter si un changement est survenu, une évaluation du meilleur individu de chaque population est comparée à sa même position précédente déjà mémorisée.

4.3.3 Pseudo-code du MBFO

```
Input: Problem_size , Cells_num , N_ed, N_re, N_c, N_s, Step_size, d_attract, w_attract, h_repellant, w_repe
P_ed, Nbr_pop, Dist_max
Populations = InitializePopulations(Cells_num, Problem_size, Nbr_pop);
Repeat
    for l = 0 to N_ed do
        for k = 0 to N_re do
            for j = 0 to N_c do
                for each population do
                    ChemotaxisAndSwim(Population, Problem_size, Cells_num, N_s, Step_size, d_attract, w_attract, h_repellant, w_repellant)
                    foreach Cell ∈ Population do
                        if Cost(Cell) <= Cost(Cell_best (population)) then
                            Cell_best (population) ← Cell;
                        end
                        if(environment changed)
                            CHANGED←true
                            go to ETQ
                        end
                    end
                end
            end
            SortByCellHealth(Population);
            Selected ← SelectByCellHealth(Population, Cells_num);
            Population ← Selected+ Selected;
        End
        for each tow populations do
            CalculateDistance
        End
        if(distance between tow populations < Dist_max)
            Randomize  population' cells of the smallest best and reevaluate it
        end
    end
    best← min(populations' Cell_best)
    ETQ
    if(CHANGED=yes)
        reevaluate (populations)
        CHANGED←false
    end
end
```

4.3.4 Expériences et résultats

4.3.4.1 La fonction de test dynamique

Branke à suggéré le problème du benchmark dynamique, appelée 'The Moving Peaks Benchmark' (**MPB**), qui consiste d'un paysage d'une fonction multidimensionnel avec plusieurs pics, dont la hauteur, la largeur et la position de chaque pic subit une petite modification chaque fois qu'un changement environnemental survient. Nous choisissons cette fonction dynamique pour construire nos problèmes de test. Cette fonction est capable de générer un certain nombre de pics en un nombre donné de dimensions qui varient à la fois dans l'espace (position et la forme du pic) et en termes de fitness. Plus de détails sur la fonction MBP est présenté dans le chapitre 2.

Paramètre	Valeur
m (nombre de pics)	10
f	Chaque 5000 Evaluations
$Height\_severity$	7.0
$Width\_severity$	1.0
$peak\_shape$	Cône
$Basic\_function$	Non
$Shift\_length\ s$	1.0
$Number\_of\_dimensions$	5
A	[0 , 100]
H	[30.0 , 70.0]
W	[1 , 12]
I	50.0

Table 4.3 : Paramètres par défaut du MBP.

La configuration des paramètres par défaut du benchmark utilisée dans les expériences peut être trouvée dans le Tableau 4.3. La longueur de déplacement ***s*** signifie qu'un pic ***p*** se déplace dans son voisinage avec un rayon ***s*** après le prochain changement de l'environnement. ***f*** désigne la fréquence de changement, ***A*** représente les valeurs minimale et maximale des limites de l'espace de recherche, ***H*** désigne la hauteur minimale et maximale des pics, ***W*** dénote les paramètres minimale et maximale de la largeur du pic et ***i*** désigne la hauteur du pic initial de tous les pics.

4.3.4.2 Métriques de performances utilisées

La performance hors ligne

La métrique classique de l'efficacité des méta-heuristiques la plus utilisée est la performance hors ligne; cependant, cette métrique est appropriée que pour les algorithmes qui évoluent une population de solutions. Quand un algorithme d'optimisation est en cours d'exécution hors-ligne, plusieurs configurations du système peuvent être évaluées (calculé le fitness) et la meilleure configuration est sélectionnée.

La performance hors-ligne qui mesure la convergence d'un algorithme est définie par l'équation qui suit, où $\bar{f}^*_{c(g)}$ est le meilleur fitness de tout membre de la population dans la génération g pour une configuration du système c. La performance hors-ligne est donc la moyenne des meilleures valeurs des fitness de chaque génération du temps courant.

$$S_c^{Horsligne} = \frac{1}{G}\sum_{g=1}^{G} \bar{f}^*_{c(g)}$$

L'erreur standard

L'erreur standard de la moyenne peut être considérée comme l'écart type de l'erreur de la moyenne des échantillons par rapport à la vraie moyenne. L'erreur standard de la moyenne est généralement estimée par l'estimateur de l'écart type divisée par la racine carrée de la taille des échantillons (n).

$$ES = \frac{s}{\sqrt{n}}$$

L'estimateur s de l'écart type utilisé est calculé comme suit:

$$s = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(X_i - \bar{X})^2}$$

Où : X_i représentent les échantillons et $\bar{X}$ est la moyenne.

4.3.5 Résultats expérimentaux

Pour tous les résultats obtenus, nous utilisons une moyenne de 50 exécutions (l'environnement change 50 fois) de l'algorithme MBFO. Les erreurs hors ligne après 5000 évaluations dans l'algorithme MBFO sont comparées à ceux de: Fast multi swarm

optimization (FMSO), Cellular PSO, Hibernating Multi Swarm Optimization (HmSO), Learning Automata based Immune Algorithm (LAIA), Cellular Differential Evolution (CDE), Adaptive Particle Swarm Optimization (APSO), Cellular PSO based on Clonal Selection (CPSOC) déjà rapporté dans [Nabizadeh 2012] et [Rezazade 2012] qui ont utilisé les mêmes expériences avec les mêmes configurations. Les tableaux 4.4, 4.5, 4.6 et 4.7 montrent l'effet du nombre de pics sur les performances du MBFO et des algorithmes précités.

M	MBFO	FMSO	Cellular PSO	HmSO	LAIA	CDE	APSO	CPSOC
1	12.00±0.55	27.58±0.94	13.46±0.7	8.53±0.49	7.34±0.32	8.20±0.19	4.81±0.14	8.29±0.55
5	8.90±0.17	19.45±0.45	9.63±0.49	7.40±0.31	7.05±0.39	6.06±0.05	4.95±0.11	6.29±0.21
10	7.23±0.10	18.26±0.32	9.42±0.21	7.56±0.27	6.91±0.32	5.93±0.04	5.16±0.11	5.45±0.17
20	7.38±0.06	17.34±0.30	8.84±0.28	7.81±0.20	6.95±0.38	5.60±0.03	5.81±0.08	5.47±0.19
30	8.40±0.05	16.39±0.48	8.81±0.24	8.33±0.18	6.92±0.33	5.56±0.03	6.03±0.07	5.59±0.12
40	7.77±0.04	15.34±0.45	8.94±0.24	8.45±0.18	6.84±0.31	5.47±0.02	6.10±0.08	5.63±0.16
50	7.91±0.05	15.54±0.26	8.62±0.23	8.83±0.17	6.43±0.29	5.47±0.02	5.95±0.06	5.74±0.11
100	9.48±0.04	12.87±0.60	8.54±0.21	8.85±0.16	6.58±0.26	5.29±0.02	6.08±0.06	5.45±0.07
200	8.39±0.03	11.52±0.61	8.28±0.18	8.85±0.16	6.41±0.27	5.07±0.02	6.20±0.04	5.79±0.10

Table 4.4 : L'erreur hors ligne et l'erreur standard pour f=500.

M	MBFO	FMSO	Cellular PSO	HmSO	LAIA	CDE	APSO	CPSOC
1	5.88±0.23	14.42±0.48	6.77±0.38	4.46±0.26	4.96±0.32	4.98±0.35	2.72±0.04	4.74±0.32
5	5.77±0.07	10.59±0.24	5.30±0.32	4.27±0.08	4.01±0.31	3.96±0.04	2.99±0.09	3.95±0.21
10	6.22±0.05	10.40±0.17	5.15±0.13	4.61±0.07	3.94±0.29	3.98±0.03	3.87±0.08	3.20±0.20
20	6.91±0.04	10.33±0.13	5.23±0.18	4.66±0.12	3.72±0.29	4.53±0.02	4.13±0.06	3.52±0.17
30	6.57±0.04	10.06±0.14	5.33±0.16	4.83±0.09	4.03±0.31	4.77±0.02	4.12±0.04	3.96±0.12
40	7.45±0.04	9.85±0.11	5.61±0.16	4.82±0.09	3.97±0.32	4.87±0.02	4.15±0.04	4.21±0.17
50	7.09±0.03	9.54±0.11	5.55±0.14	4.96±0.03	4.22±0.31	4.87±0.02	4.11±0.03	3.98±0.11
100	7.63±0.03	8.77±0.09	5.57±0.12	5.14±0.08	4.19±0.32	4.85±0.02	4.26±0.04	4.13±0.12
200	8.73±0.04	8.06±0.07	5.50±0.12	5.25±0.08	4.38±0.31	4.46±0.01	4.21±0.02	4.15±0.01

Table 4.5 : L'erreur hors ligne et l'erreur standard pour f=1000.

M	MBFO	FMSO	Cellular PSO	HmSO	LAIA	CDE	APSO	CPSO C
1	4.67+0.18	6.29±0.20	4.15±0.25	1.75±0.10	2.48±0.15	2.38±0.78	1.06±0.03	2.31±0.21
5	5.76+0.06	5.03±0.12	2.85±0.24	1.92±0.11	2.51±0.19	2.12±0.02	1.55±0.05	2.01±0.13
10	5.91+0.04	5.09±0.09	2.80±0.10	2.39±0.16	2.82±0.27	2.42±0.02	2.17±0.07	1.56±0.15
20	6.32+0.04	5.32±0.08	3.41±0.14	2.46±0.09	3.16±0.36	3.05±0.04	2.51±0.05	2.41±0.13
30	5.82+0.03	5.22±0.08	3.62±0.12	2.57±0.05	3.14±0.33	3.29±0.03	2.61±0.02	2.78±0.10
40	5.96+0.03	5.09±0.06	3.84±0.12	2.56±0.06	3.02±0.31	3.43±0.03	2.59±0.03	2.90±0.12
50	5.90+0.03	4.99±0.06	3.86±0.10	2.65±0.05	3.05±0.31	3.44±0.02	2.66±0.02	3.18±0.09
100	6.03+0.02	4.60±0.05	4.10±0.11	2.72±0.04	3.14±0.35	3.36±0.01	2.62±0.02	3.22±0.07
200	5.89+0.02	4.34±0.04	3.97±0.10	2.81±0.04	3.08±0.32	3.13±0.01	2.64±0.01	3.09±0.12

Table 4.6 : L'erreur hors ligne et l'erreur standard pour f=2500.

M	MBFO	FMSO	Cellular PSO	HmSO	LAIA	CDE	APSO	CPSO C
1	3.63±0.05	3.44±0.11	2.55±0.12	0.87±0.05	1.94±0.19	1.53±0.07	0.53±0.01	1.02±0.14
5	3.85±0.03	2.94±0.07	1.68±0.11	1.18±0.04	2.09±0.18	1.50±0.04	1.05±0.06	0.99±0.15
10	4.05±0.03	3.11±0.06	1.78±0.05	1.42±0.04	2.14±0.15	1.64±0.03	1.31±0.03	1.75±0.10
20	4.11±0.02	3.36±0.06	2.61±0.07	1.50±0.06	2.97±0.21	2.64±0.05	1.69±0.05	1.93±0.11
30	3.99±0.02	3.28±0.05	2.93±0.08	1.65±0.04	2.98±0.23	2.62±0.05	1.78±0.02	2.28±0.10
40	4.36±0.02	3.26±0.04	3.14±0.08	1.65±0.05	3.07±0.29	2.76±0.05	1.86±0.02	2.62±0.09
50	4.97±0.02	3.22±0.05	3.26±0.08	1.66±0.02	2.93±0.27	2.75±0.05	1.95±0.02	2.74±0.10
100	5.64±0.03	3.06±0.04	3.41±0.07	1.68±0.03	3.06±0.24	2.73±0.03	1.95±0.01	2.84±0.12
200	6.38±0.03	2.84±0.03	3.40±0.06	1.71±0.02	2.95±0.23	2.61±0.02	1.90±0.01	2.69±0.08

Table 4.7 : L'erreur hors ligne et l'erreur standard pour f=5000.

4.3.6 Discussion et analyse des résultats

A travers la comparaison des résultats de notre algorithme à ceux des autres algorithmes, le tableau 4.4 montre les mesures de l'erreur hors ligne pour une fréquence de changement de l'environnement très élevée (f= 500), MBFO montre de bonnes performances pour un tel environnement par rapport à plusieurs algorithmes; pour des fréquences de changement moins élevées les performances de notre algorithme commence à diminuer, par exemple : dans le tableau 4.5 et pour (f=1000) MBFO ne dépasse significativement que l'algorithme FMSO. Quand à l'erreur standard, les différents résultats montrent que notre

algorithme MBFO obtient de meilleurs résultats dans la majorité des tests par rapport aux autres algorithmes et pour les différentes fréquences de changement de l'environnement (500, 1000, 2500, 5000).

4.3.7 Conclusion

Dans ce travail et pour la première fois un nouvel algorithme d'optimisation multi-population basé sur le comportement intelligent des bactéries a été décrit. Les expériences ont été faites sur le benchmark MBP et les résultats de la méthode proposée ont été comparé à plusieurs méthodes de la littérature. Bien que les résultats expérimentaux obtenus à traves des études comparatives en terme de l'erreur hors ligne ne montrent pas la supériorité de notre algorithme proposé MBFO dans le cas général, MBFO obtient de meilleurs résultats dans des environnements caractérisés par un changement rapide. On outre cet algorithme montre une convergence rapide et une grande capacité de localiser les différents pics et de les poursuivre ainsi qu'une bonne stabilité interprétée par l'erreur standard très minimale dû au bon mécanisme de la diversité adopté par cet algorithme.

Chapitre 5

Conclusion et perspectives

5.1 Conclusion

Beaucoup de problèmes d'optimisation réels changent avec le temps (dynamiques), tels que la poursuite des cibles, la planification des tâches… etc. Une façon de faire face au changement est de considérer le problème comme nouveau et lancer la recherche à partir de zéro. Toutefois, si le nouvel optimum n'est pas trop loin de l'ancien, l'information acquise pourrait être utile simplement en adaptant le processus de recherche sur les changements environnementaux et sans redémarrer le processus d'optimisation. C'est pourquoi un algorithme d'optimisation dynamique doit non seulement essayer de trouver la meilleure solution (qui peut changer après qu'elle soit atteinte ou même durant le processus de sa recherche), mais il doit le suivre. Par conséquent, des techniques adaptatives et efficaces ont été proposées pour trouver une nouvelle solution aussi vite que possible dans un environnement dynamique.

Dans notre mémoire qui traite le problème de l'optimisation dans les environnements dynamiques, nous avons présenté les différentes méthodes d'optimisation dynamique par des approches inspirées de la nature et un état de l'art sur les principaux travaux menés dans ce domaine ainsi que les différentes techniques utilisées pour améliorer et adapter ces approches classiques (EA, PSO, ACO, etc ...) afin de résoudre ce problème (introduire la diversité après un changement, le maintien de la diversité, utiliser la mémoire, la prédiction et utiliser plusieurs populations). Dans notre travail, nous avons choisi l'approche BFO dont sa philosophie semble être facilement et mieux adaptée aux environnements dynamiques par rapport à nombreuses approches, où le maintien de la diversité est assuré par les étapes d'élimination et de la dispersion pour éviter la convergence prématurée durant l'ensemble du processus de recherche. La Chimiotaxie et les étapes de la reproduction assurent une bonne exploitation (recherche locale) et une convergence rapide.

Dans un premier temps, nous avons utilisé BFO avec plusieurs techniques auto-adaptatives de diversité et de prédiction pour résoudre le problème dynamique de la poursuite d'objets. Les résultats obtenus montrent l'efficacité de ces techniques appliquées dans toutes les expériences. Dans un deuxième temps nous avons utilisé BFO et pour la première fois en tant que plusieurs populations. Dans cette version multi-population nous avons introduit un mécanisme d'exclusion entre les populations et un mécanisme de réactivité aux changements pour une bonne exploration/exploitation de l'espace de recherche. Les performances de notre algorithme ont été testées sur le benchmark MBP. Les résultats montrent la compétitivité de cette approche par rapport aux autres approches dans des fréquences de changement élevés.

5.2 Perspectives

Le dynamisme est présent de plus en plus dans les différents problèmes du monde réel et il est nécessaire de trouver des approches plus efficaces pour traiter ces problèmes. Les méthodes classiques telles que le redémarrage du processus d'optimisation après chaque changement de l'environnement ne sont plus une option. Résoudre des problèmes dynamiques exige le traitement de plusieurs questions liées au développement des algorithmes, à la mise en œuvre et aux tests. Bien qu'il existe plusieurs techniques introduites aux approches classiques pour les adapter aux problèmes dynamiques, les approches qui utilisent plusieurs populations, les approches de l'auto-adaptation et les approches qui utilisent un bon mécanisme de diversité sont les méthodes les plus utilisées pour traiter la majorité des problèmes dynamiques.

Malgré que plusieurs études ont été réalisées pour traiter le problème de l'optimisation dynamique, la plus part de ces travaux visent uniquement des problèmes continues et il n'existe que très peu de travaux sur l'optimisation dynamique combinatoire et cela est dû principalement au manque des benchmarks dynamiques standards des différents problèmes combinatoires connus et leurs difficultés par rapport aux problèmes continus.

Bibliographie

[Baluja 1994] Baluja S. (1994). "*Population-Based Incremental Learning - A Method for Integrating Genetic Search Based Function Optimization and Competitive Learning*". Tech. report CMU-CS-94-163, Computer Science Department, Carnegie Mellon University.

[Barlow 2011] Barlow G. J. (2011). "*Improving memory for optimization and learning in dynamic environments*". Doctoral dissertation, tech. report CMU-RI-TR-11-17, Robotics Institute, Carnegie Mellon University.

[Blackwell 2002] Blackwell T.M., Bentley P. J. (2002). "*Dynamic Search with Charged Swarms*". GECCO 02 Proceedings of the Genetic and Evolutionary Computation Conference, pp. 19-26.

[Bosman 2005] Bosman P. A.N. (2005). "*Learning, Anticipation and Time–Deception in Evolutionary Online Dynamic Optimization*". GECCO'05, June 25–29, Washington, DC, USA.

[Branke 1999] Branke J. (1999). "*Memory Enhanced Evolutionary Algorithms for Changing Optimization Problems*". Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress, Vol. 3, pp. 1875-1882.

[Branke 2000] Branke J., Kaußler T., Schmidt C., and Schmeck H. (2000). "*A Multi-Population Approach to Dynamic Optimization Problems SOS*". Adaptive Computing in Design and Manufacture, pp. 299-308.

[Branke 2004] Branke J. (2004). "*Optimization in Dynamic Environments*". GECCO Tutorial. Institute AIFB, University of Karlsruhe Germany.

[Bui 2005] Bui L. T., Branke J. and Abbass H. A. (2005). "*Multiobjective optimization for dynamic environments*". Evolutionary Computation, The 2005 IEEE Congress, Vol. 3, pp. 2349 - 2356

[Brownlee 2011] Brownlee J. "*Clever Algorithms: Nature-Inspired Programming Recipes*". Book, First Edition, LuLu, ISBN: 978-1-4467-8506-5.

[Cruz 2011] Cruz Carlos, Gonzalez J. R., and Pelta D. A. (2011). "*Optimization in dynamic environments: A survey on problems, methods and measures*". Published in Soft Computing, 15(7), pp. 1427-1448.

[Chuan 2007] Chuan L. and Quanyuan F. (2007). "*A Hierarchical Subpopulation Particle Swarm Optimization Algorithm*". ISKE-2007 Proceedings Advances in Intelligent Systems Research.

[Dan 2009] Dan S. (2009). "*An Analysis of Diploidy and Dominance in Genetic Algorithms*". International Conference on Computer, Communication, Control and Information Technology, West Bengal, India.

[Daneshyari 2011] Daneshyari M. and Yen G. G. (2011). "*Dynamic Optimization using Cultural based PSO*". Evolutionary Computation (CEC), 2011 IEEE Congress.

[Dizdar 2011] Dizdar D., Gershkov A., and Moldovanu B. (2011). "*Revenue maximization in the dynamic knapsack problem*". Theoretical economics: TE, journal of the Econometric Society, open access to research in economic theory, Toronto, Vol. 6, pp. 157-184

[Dorigo 2003] Dorigo M. and Stützle T. (2003). "*Ant Colony Optimization ACO*". Library of Congress Cataloging in Publication Data, ISBN 0-262-04219-3.

[Eberhart 1995] Eberhart R. (1995). "*A new optimizer using particle swarm theory*". Micro Machine and Human Science, MHS 95, Proceedings of the Sixth International Symposium.

[Egmond 2012] Egmond S. V. (2012). "*Dynamic Ant Colony Optimization for the Traveling Salesman Problem*". Thesis, Leiden Institute of Advanced Computer Science (LIACS) Leiden University Niels Bohrweg 12333 CA Leiden The Netherlands.

[Eisuke 2011] Eisuke Kita. (2011). "*Evolutionary algorithms*". Book, InTech Janeza Trdine 9, 51000 Rijeka, Croatia, ISBN 978-953-307-171-8.

[Eyckelhof 2002] Eyckelhof C. J. and Snoek M. (2002). "*Ant Systems for a Dynamic TSP Ants caught in a traffic jam*". Ant Algorithms, Third International Workshop, ANTS 2002, Brussels, Belgium, September 12-14, 2002, Proceedings, Vol. 2463, pp. 88-99

[Goh 2009] Goh C. K. and Tan K. C. (2009). "*A Competitive-Cooperative Coevolutionary Paradigm for Dynamic Multiobjective Optimization*". IEEE transactions on evolutionary computation, vol. 13, no. 1, pp. 103-127

[Grefenstette 1992] Grefenstette J. J. (1992). "*Genetic algorithms for changing environments*". 2nd Int. Conf. on Parallel Problem Solving from Nature, pp. 137-144.

[Guntsch 2001] Guntsch M., Middendorf M., and Schmeck H. (2001). "*An Ant Colony optimization Approach to Dynamic TSP*". Proceedings of the Genetic and Evolutionary Computation Conference (GECCO) 2001, pp. 860-867

[COBB 1990] COBB H. G. (1990). "*An Investigation into the Use of Hypermutation as an Adaptive Operator in Genetic Algorithms Having Continuous, Time-Dependent Nonstationary Environments*". NRL - MR 6760 United States.

[De Castro 2002] De Castro L. N. and Von Zuben F. J. (2002). "*Learning and Optimization Using the Clonal Selection Principle*". IEEE transactions on evolutionary computation, Vol. 6, no. 3, pp. 239-251

[Hassan 2004] Hassan R., Cohanim B.and De Weck O. "A Copmarison of Particle Swarm Optimization and the Genetic Algorithm". American Institute of Aeronautics and Astronautics

[Hendtlass 2009] Hendtlass T., Moserand I. and Randall M. (2009). "*Dynamic Problems and Nature Inspired Meta-Heuristics*". A. Lewis et al. (Eds.): Biologically-Inspired Optimisation Methods, SCI 210, pp. 79-109

[Iceko 2003] ICeko H., Skok M., and Skrlec D. (2003). "*Artificial Immune Systems in Solving Routing Problems*". EUROCON 2003, Computer as a Tool. The IEEE Region 8, Vol. 1, pp. 62 - 66

[Janson 2006] Janson S. and Middendorf M. (2006). "*A hierarchical particle swarm optimizer for noisy and dynamic environments*". Genet Program Evolvable Mach. Springer Science+Business Media, LLC, Vol. 7 no. 4, pp. 329-354

[Klein 2013] Klein D. A. (2013). "*BoBoT - Bonn Benchmark on Tracking,"*. www.iai.uni-bonn.de/~kleind/tracking/. Consulted on 01-05-2013

[Liu 2008] Liu L., Wang D. and Yang S. (2008). "*Compound Particle Swarm Optimization in Dynamic Environments*". EvoWorkshops 2008: EvoCOMNET, EvoFIN, EvoHOT, EvoIASP, EvoMUSART, EvoNUM,

EvoSTOC, and EvoTransLog, Naples, Italy, March 26-28, 2008. Proceedings , pp. 616-625

[Liu 2012] LIU L., WANG D. and TANG J. (2012). "*Immune system based differential evolution algorithm using near-neighbor effect in dynamic environments*". Journal of Control Theory and Applications, Vol. 10, no 4, pp. 417-425

[Mori 1998] Mori N., Kita H. and Nishikawa Y. (1998). "*Adaptation to a Changing Environment by Means of the Feedback Thermodynamical Genetic Algorithm*". Parallel Problem Solving from Nature - PPSN V Lecture Notes in Computer Science Vol. 1498, pp. 149-158

[Morrison 1999] Morrison R.W, De Jong K.A. (1999). "*A Test Problem Generator for Non-Stationary Environments*". Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress, Vol. 3, pp. 2047-2053.

[Nabizadeh 2012] Nabizadeh S., Rezvanian A. and M. Meybodi R. (2012). "*A Multi-Swarm Cellular PSO based on Clonal Selection Algorithm in Dynamic Environments*". IEEE/OSA/IAPR International Conference on infonnatics, Electronics & Vision, pp. 482-486

[Nguyen 2012] Nguyen, Trung T., Yang S., and Branke J. (2012). "*Evolutionary dynamic optimization: A survey of the state of the art*". Swarm and Evolutionary Computation, Vol. 6, pp. 1-24

[Olariu 2006] Olariu S. and Zomaya A. Y. (2006). "*Handbook of Bioinspired Algorithms and Applications*". Chapman & Hall/CRC Taylor & Francis Group.

[Oppacher 1999] Oppacher F. and Wineberg M. (1999). "*The Shifting Balance Genetic Algorithm: Improving the GA in a Dynamic Environment*". Genetic and Evolutionary Computation Conference, vol. 1, pp. 504-510.

[Ouerfelli 2001] Ouerfelli L. and Bouziri H. (2011). "*Greedy algorithms for dynamic graph coloring*". Communications, Computing and Control Applications (CCCA), International Conference, pp. 1-5.

[Parrott 2006] Parrott D. and Li X. (2006). "*Locating and Tracking Multiple Dynamic Optima by a Particle Swarm Model Using Speciation*". IEEE transactions on evolutionary computation, Vol. 10, no. 4, pp. 440-458

[Passino 2002] Passino K. M. “Biomimicry of bacterial foraging”. IEEE Control Systems Magazine, pp. 52–67

[Pinto 2013] Pinto P. C., Runkler T. A. and Sousa J. M.C. (2013). "*Insect Swarm Algorithms for Dynamic MAX-SAT Problems*". Book, Metaheuristics for Dynamic Optimization Studies in Computational Intelligence, Springer Berlin Heidelberg, pp. 341-369

[Peng 2010] Peng X., Gao X. and Yang S. (2010). "*Environment identification-based memory scheme for estimation of distribution algorithms in dynamic environments*". Soft Computing Vol. 15, no. 2, pp. 311-326

[Rainer 1997] Rainer S. and Kenneth P. (1997). "*Differential Evolution: A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces*". Journal of Global Optimization, Vol. 11, no. 4, pp. 341-359

[Ramsey 1993] Ramsey C. L. and Grefenstette J. J. (1993). "*Case-Based Initialization of Genetic Algorithms*". In Stephanie Forrest, editor, Proceedings of Fifth International Conference on Genetic Algorithms, pp. 84-91.

[Rezazade 2012] Rezazade I., Ghatei S. and Naebi A. (2012). "*A Modified Particle Swarm*

Optimization Using FCM for Moving Peaks Benchmark". Journal of Basic and Applied Scientific Research, Vol.2 pp.4266-4274

[Riekert 2009] Riekert M., Malan K. M. and Engelbrect A.P. (2009). "*Adaptive Genetic Programming for Dynamic Classification Problems*". Evolutionary Computation, CEC '09. IEEE Congress , pp. 674 - 681

[Rossi 2008] Rossi C., Abderrahim M. and Diaz J.C. (2008). "*Tracking Moving Optima Using Kalman-Based Predictions*". Evolutionary Computation journal, Vol. 16, no. 1, pp. 1-30

[Sedighizadeh 2009] Sedighizadeh D. and Masehian E. (2009). "*Particle Swarm Optimization Methods, Taxonomy and Applications*". International Journal of Computer Theoryand Engineering, Vol. 1, no. 5, pp. 1793-8201

[Simoes 2008] Simoes A. and Costa E. (2008). "*Evolutionary Algorithms for Dynamic Environments: Prediction Using Linear Regression and Markov Chains*". 10th International Conference Dortmund, Germany, September 13-17, 2008 Proceedings, pp 306-315

[Sjoerd 2012] Sjoerd V. E. (2012). "*Dynamic Ant Colony Optimizationf or the Traveling Salesman Problem*". master's thesis, Leiden Institute of Advanced Computer Science (LIACS) Leiden University Niels Bohrweg 1 2333 CA Leiden The Netherlands.

[Tang 2006] Tang W. J., Wu Q. H. and Saunders J. R. (2006). "*Bacterial Foraging Algorithm For Dynamic Environments*". 2006 IEEE Congress on Evolutionary Computation Sheraton Vancouver Wall Centre Hotel, Vancouver, BC, Canada, pp. 16-21

[Vavak 1998] Vavak F., Jukes K. A. and Fogarty T. C. (1998). "*Performance of a Genetic Algorithm with Variable Local Search Range Relative to Frequency of the Environmental Changes*". Genetic Programming: Proceedings of the Third Annual Conference, pp. 602-608

[Wang 2006] Wang Y. and Wineberg M. (2006). "*Estimation of evolvability genetic algorithm and dynamic environments*". Genetic Programming and Evolvable Machines, Vol. 7, no. 4, pp. 355-382

[Weise 2009] Weise T. "*Global Optimization Algorithms - Theory and Application*". http://www.it-weise.de/. Consulted on 23-01-2013

[Yang 2005] Yang S. and Yao X. (2005). "*Experimental Study on Population-Based Incremental Learning Algorithms for Dynamic Optimization Problems*". Soft Computing November 2005, Vol. 9, no. 11, pp. 815-834

[Yang 2010] Yang X.S. (2010). "*Firefly Algorithm, Levy Flights and Global Optimization*". Book, Incorporating Applications and Innovations in Intelligent Systems XVII, pp. 209-218

[Zhou 2007] Zhou A., Jin Y., Zhang Q., Sendhoff B. and Tsang E. (2007). "*Prediction-Based Population Reinitialization for Evolutionary Dynamic Multi-objective Optimization*". Evolutionary Multi-Criterion Optimization Lecture Notes in Computer Science, Vol. 4403, pp. 832-846

Annexe 1

A Dynamic Auto-Adaptive Bacterial Foraging Optimization for Color-Based Object Tracking". Proceeding of the 2nd International Conference on Segnal, Image, Vision and their Applications (SIVA'13), November 18-20, 2013 - Guelma, Algeria, pp. 413-418

GUELMA UNIVERSITY

PI:MIS
Laboratory

International Conference on

Signal, Image, Vision and their Applications SIVA'13

November 18-20, 2013, Guelma

Inverse Problems Modeling, Information and Systems Laboratory (PI MIS)
Faculty of Sciences and Technology, Guelma University, Guelma 24000, Algeria

Preface

The May 8th, 1945 University and the PI:MIS laboratory are very pleased to welcome you in Guelma at the 2nd International Conference on Signal, Image, Vision and their applications SIVA13.

The SIVA13 is an ideal "culture broth" for new ideas, contacts and collaborations both for confirmed scientists as well as younger scientists and students just starting out in this exciting and rapidly changing field of signal, image and vision discovery research and development.

A wide range of topics is addressed, including biomedical signal and image processing, image and video processing, biometrics, image segmentation and characterization, image filtering and restoration, image and signal compression, texture analysis, pattern recognition, design and implementation of signal processing systems, signal processing for communications systems, system identification, modeling and simulation, inverse problems.

I deeply thank all those who by their devotion, patience, willingness, competence, dynamism and their good mood, allowed the realization and the good progress of these three days of the SIVA13.

We are delighted that the invited speakers are very impressive, each of them having an excellent track record in his own field of research. I would like to thank the speakers and all the participants who insured the animation of the conference. I hope that these three days will answer your expectations and that the exchanges as well as the discussions will be fruitful and profitable to everybody.

It is thus with great pleasure that we welcome you in Guelma for what we hope will be an extremely rewarding and memorable meeting for everyone!.

Prof. Abdelhani BOUKROUCHE
PI: MIS Laboratory

UNIVERSITÉ 8 MAI 1945 GUELMA

International Conference on Signal, Image, Vision and their Applications SIVA'13

© SIVA'13, November 18-20, 2013, Guelma Algeria

Organizer

Inverse Problems Modeling, Information and Systems Laboratory (PI:MIS)

Organizing Committee

President: Abdelhani BOUKROUCHE

Members:

Smain SAHOUR, Med Salah BOUDELIOUA, Abdelaziz YOUNSI, Houcine BOUROUBA, Mohamed Cherif AMARA KORBA, Hakim DOGHMANE, Nadjib BOUSSETILA, Mohamed NEMISSI,Salim BENDJOUDI, Houria BOUDOUDA, Soraya ZENATI, Djalel DRICI, El Hadi MEHALLEL, Lotfi HOUAM, Laatra YOUSFI, Amir BENZAOUI, Nabil HEZIL, Hidjaz HEZIL, Badreddine GUERROUI, Fethi CHOUAF, Abdessadek SAIB, Mourad BENZAHI.

Scientific Committee

President: R. JENNANE (France)

Members:

K. ABED-MERAIM (France),Sos S. AGAIAN (USA), A. ALMHDIE (France), M. C. AMARA KORBA (Algeria), N. AMARDJIA (Algeria), M. BEDDA (Saudi Arabia), M. Hedi BEDOUI (Tunisia), A. BELATRECHE (United Kingdom), A. BENIA (Algeria), Mohamed BENOUARET (Algeria), N. BETROUNI France), S. BOUAKEZ (France), M. S. BOUDELLIOUA (Oman),T. BOUDEN (Algeria), H. BOUDOUDA (Algeria), A. BOUKHAROUBA (Algeria), A. BOUKROUCHE (Algeria),H. BOURDOUCEN (Oman), A. BOURIDANE (United Kingdom), H. BOUROUBA (Algeria), N. BOUSSETILA (Algeria),A. BOUZERDOUM (Australia), Fabio Martinez CARRILLO (Colombia), M. CHADLI (France),A. CHEDDAD (Sweden), A. F. CHAREF (Algeria), Aladine CHETOUANI (France),Amina CHANTIR (Algeria), Ke. CHEN (United Kingdom),K. CURRAN (Northen Ireland), M. DERICHE (Saudi Arabia),K. DJEMAL (France), R. DJIMILI (Algeria), N. DOGHMANE (Algeria),W. ELLOUMI (France), M. Fezari (Algeria),M. GABBOUJ (Finland), A. GANOUN (Libya),A. GUESSOUM (Algeria), A. HADID (Finland), A. HAFIANE (France), K. HAMROUNI (Tunisia), W. HAMOU-LHADJ (Canada), A. HASSINI (Algeria), M. EL HASSOUNI (Morocco), M. F. HARKET (Algeria),L. HOUAM (Algeria),M. JABLOUN (France), R. KACHOURI (France), I. KACHOURI (France), M. KEDIR (Algeria), M. T. KHADIR (Algeria), M. KHAMADJA (Algeria),K. KHELIL (Algeria), M. N. KOUAHLA (Algeria), Y. LAFIFI (Algeria), R. LEDEE (France),B. MAHRIC (France), A. MANZANERA (France), A. MARCHADIER (France), S. MARCHAD-MAILLET (Switzerland), A. MITICHE (Canada), A. MOUSSAOUI (Algeria),A. NAIT-ALI (France), M. NEMISSI (Algeria),M. OUSSALAH (United Kingdom), Ph. NEVEUX (France), P. RAVIER (France), S. SAHOUR (Algeria),Frederic ROS (France), D. SBIBIH (Morocco), Y. SMARA (Algeria), H. SERIDI (Algeria),A. SERIR (Algeria), A. TALEB-AHMED (France), Ch. TOLBA (Algeria), O. UCAN (Turkey), L. YOUSFI (Algeria), S. ZENATI (Algeria).

A Dynamic Auto-Adaptive Bacterial Foraging Optimization for Color-Based Object Tracking

Mohamed Skander Daas
Department of mathematics and computer science
University Oum El Bouaghi
Oum El-Bouaghi, Algeria
daas.skander@live.com

Meriem Bettoum
Department of computer science
University Constantine 2
Constantine, Algeria
meryam.bettoum@gmail.com

Maroua Mahmoudi
Department of computer science
University Constantine 2
Constantine, Algeria
maroua2mahmoudi@gmail.com

Mohamed Batouche
Department of computer science
University Constantine 2
Constantine, Algeria
batouche@ccis.edu.sa

ABSTRACT

Object tracking is an active research area and a challenging problem that can be formulated as a dynamic optimization problem. In this paper we present a dynamic Bacterial Foraging Optimization algorithm (DBFO) to solve object tracking problem using histograms similarity. This basic dynamic version is extended to an auto-adaptive one (DABFO) by introducing some adaptive techniques of prediction and diversity maintaining which are based on trajectory, velocity and evolution of the best fitness. The proposed techniques ensure a good diversity and more efficient exploration-exploitation trade-off of the search space in a dynamic environment. A color-based method is used to define the bacteria fitness, in which it is calculated using Hue level histogram of the HSV color space. Experimental results have shown that both DBFO and DABFO perform well in object tracking problem. Moreover, the proposed DABFO competes and even outperforms existing algorithms based on BFO especially when severity and velocity changes are large and not stable.

Keywords: *component; object tracking; bacterial foraging optimization; color histograms similarity; dynamic optimization; auto-adaptive prediction; diversity;*

I. INTRODUCTION

While some object characteristics like shape, rotation …etc are severely changing in a noisy environment, using color space-based tracking method is a good choice comparing to other methods like region-based tracking, active contour-based tracking, and model-based tracking. In color space-based tracking method, tracking algorithms utilize information available from the color spaces to develop metrics for tracking purposes such as our selected technique (Color Histograms Similarity), where histograms change slowly under change of angle of view, scale and occlusion. This technique utilizes a selected rectangular reference region for tracking within an image frame. The target object can be tracked by a correlation matching between the reference image region window and a candidate sub region window of the frame image, where it quantifies the similarity of the two groups of pixel values.

Object tracking can be considered like a dynamic optimization problem (DOP) because of both changing over time of environment and target object position, one way to cope with change is to consider the problem as a new and start the search from scratch. However, if the new optimum is not too far from the old one, the gained information could be useful by just adapting the search process to the environmental changes and without restarting it. That is why a dynamic optimization algorithm should not only try to find the best solution (that can change after it was reached or even during the process of its research), but it should track it, therefore, adaptive and efficient techniques should be used to find the new solution as quickly as possible.

In [1] different techniques are presented to enhance and adapt traditional metaheuristics (EA, PSO, ACO …etc) to solve DOPs (Diversity introducing, Diversity maintaining, Memory, Prediction and Multi-Population). In our work we have chosen BFO approach in which its philosophy seems to be easily and better adapted to dynamic environments than many approaches, where diversity maintaining is insured by elimination and dispersal steps to avoid premature convergence during the whole search process, chemotaxis and reproduction steps insure a good exploitation (local search) and fast convergence.

It has been known that other evolutionary computation algorithms like GA and PSO were developed initially for static optimization problems where it shows good performance, on the other hand, these algorithms suffer from several problems in dynamic environments (premature convergence, diversity loss, memory expiration …etc) [2]. For example, unlike BFO, PSO algorithm individuals behavior is directly influenced by their L_{best} and G_{best} positions, and these two positions' costs are changing over time, therefore, the algorithm results a bad interpretation of these two parameters, in addition with PSO the search space will not be covered enough after convergence.

In [3] a Dynamic BFA has been proposed for the first time, it is worth mentioning that the diversity of DBFA changes after each Chemotaxis process rather than the dispersion adopted by the BFA after several generations, the DBFA utilizes not only the local search but also applies a flexible selection scheme (roulette wheel selection) to maintain a suitable diversity during the whole evolutionary process and it has shown that it outperforms BFA in almost all dynamic environments. Recently in [4], a modified bacterial foraging optimization algorithm (m-BFO) is presented, and shows that it outperforms

PSO in a number of important metrics for pedestrian tracking. Also the proposed BFO approach is compared with other commonly used trackers, where the BFO and PSO swarm intelligence approaches achieve comparable results to the more often used particle filter. Other works based on histograms similarity are presented in [9][10][11][12] [13].

In this paper we present a dynamic version of BFO (similar to m-BFO) adapted to environment changes called DBFO which is extended to an auto-adaptive version (DABFO) by applying some of DOPs techniques (memory, prediction and diversity maintaining), so the main improvement of this work is the auto-adaptive techniques used by DABFO which improve its accuracy performance.

Regarding experiments, some video benchmarks are used from the BoBOT benchmark [8]. To evaluate our algorithms, we have used The Accuracy Performance and the Collective Mean Fitness measures.

The rest of this paper is organized as follows: in section II the BFO approach is described, section III provides a color histograms similarity description and the fitness function formulation, the DABFO algorithm is described in section IV, in section V different experiments are presented with a discussion of results, and finally a conclusion is drawn in section VI.

II. BACTERIAL FORAGING OPTIMIZATION

The Bacterial Foraging Optimization Algorithm presented in [9] belongs to the field of Bacteria Optimization Algorithms and Swarm Optimization, and more broadly to the fields of Computational Intelligence and Metaheuristics. It is related to other Bacteria Optimization Algorithms such as the Bacteria Chemotaxis Algorithm, and other Swarm Intelligence algorithms such as Ant Colony Optimization and Particle Swarm Optimization [5].

The information processing strategy of the algorithm is to allow cells to stochastically and collectively swarm toward optima. This is achieved through a series of three processes on a population of simulated cells:

1) *Chemotaxis:* where the cost of cells is derated by the proximity to other cells and cells move along the manipulated cost surface one at a time.

2) *Reproduction:* where only those cells that are well performed over their lifetime may contribute to the next generation.

3) *Elimination-dispersal:* where cells are discarded and new random samples are inserted.

A bacteria cost is reduced by its interaction with other cells. This interaction function g is calculated as follows:

$$g(cell_k)= \sum_{i=1}^{S}\left[-d_{attr}\times\exp\left(-w_{attr}\times\sum_{m=1}^{P}(cell_m^k-other_m^i)^2\right)\right] + \sum_{i=1}^{S}\left[d_{repel}\times\exp\left(-w_{repel}\times\sum_{m=1}^{P}(cell_m^k-other_m^i)^2\right)\right] \quad (1)$$

Where $cell_k$ is a given cell, d_{attr} and w_{attr} are attraction coefficients, h_{repel} and w_{repel} are repulsion coefficients, S is the number of cells in the population, P is the number of dimensions on a given cells position vector.

The remaining parameters of the algorithm are as follows $Cells_{num}$ is the number of cells maintained in the population, N_{ed} is the number of elimination and dispersal steps, N_{re} is the number of reproduction steps, N_c is the number of chemotaxis steps, N_s is the number of swim steps for a given cell, $Step_{size}$ is a random direction vector with the same number of dimensions as the problem space, and each value $\in [-1,1]$, and P_{ed} is the probability of a cell being subjected to elimination and dispersal.

A. Pseudo code for the BFO algorithm:

Input: $Problem_{size}$, $Cells_{num}$, N_{ed}, N_{re}, N_c, N_s, $Step_{size}$, $d_{attract}$, $w_{attract}$, $h_{repellant}$, $w_{repellant}$, P_{ed}
Output: $Cell_{best}$
Population ← InitializePopulation($Cells_{num}$, $Problem_{size}$);
for l = 0 to N_{ed} do
  for k = 0 to N_{re} do
    for j = 0 to N_c do
      ChemotaxisAndSwim(Population, $Problem_{size}$, $Cells_{num}$, N_s, $Step_{size}$, $d_{attract}$, $w_{attract}$, $h_{repellant}$, $w_{repellant}$);
      foreach *Cell* □ Population do
        if Cost(*Cell*) ≤ Cost($Cell_{best}$) then
          $Cell_{best}$ ← *Cell*;
        end
      end
    end
    SortByCellHealth(Population);
    Selected ← SelectByCellHealth(Population, $Cells_{num}$);
    Population ← Selected+ Selected;
  end
  foreach *Cell* □ Population do
    if Rand() ≤ P_{ed} then
      Cell ← CreateCellAtRandomLocation();
    end
  end
end
return $Cell_{best}$;

B. Pseudocode for ChemotaxisAndSwim function:

Input: Population, $Problem_{size}$, $Cells_{num}$, N_s, $Step_{size}$, $d_{attract}$, $w_{attract}$, $h_{repellant}$, $w_{repellant}$
foreach *Cell* □ Population do
  $Cell_{fitness}$ ← Cost(*Cell*) + Interaction(*Cell*, Population, $d_{attract}$, $w_{attract}$, $h_{repellant}$, $w_{repellant}$);
  $Cell_{health}$ ← $Cell_{fitness}$;
  Cell'← □;
  for $i = 0$ to N_s do
    RandomStepDirection ← CreateStep($Problem_{size}$);
    Cell'←TakeStep(RandomStepDirection, $Step_{size}$);
    $Cell'_{fitness}$←Cost(*Cell'*)+Interaction(*Cell'*, Population, $d_{attract}$, $w_{attract}$, $h_{repellant}$, $w_{repellant}$);
    if $Cell'_{fitness} > Cell_{fitness}$ then
      $i \leftarrow N_s$;
    else
      $Cell \leftarrow Cell'$;
      $Cell_{health} \leftarrow Cell_{health} + Cell'_{fitness}$;
    end
  end
end

III. OBJECT TRACKING USING HISTOGRAMS SIMILARITY

A. Color histogram for object tracking:

Color is widely used for describing object due to its invariance of translation, rotation and scale [6]. *YCbCr* and HSV color spaces had consistently higher tracking performance compared to grayscale color space, tracking results also indicate that utilization of a single layer where the color information was stored could be used for tracking [7]. The color information in the *YCbCr* color space are stored in the *Cb* and *Cr* layers whereas in the HSV color space are stored in the hue (*H*) and saturation (*S*) Layers, The hue of a color refers to the spectral wavelength that closely matches to the colors of the rainbow. Typically, a hue of 0° is red, which also is used as the reference value, 120° is green and 240° is blue. The saturation (*S*) parameter is the measure of the purity of the color, the value (*V*) parameter indicates the brightness levels or the grayscale values that range from black to white.

Unlike RGB, HSV separates Luma (the image intensity), from Chroma (the color information); this is very useful in color based tracking. In our work we use Hue color space histograms similarity of the HSV color space.

An image histogram is a type of histogram that acts as a graphical representation of the tonal distribution in a digital image. It plots the number of pixels for each tonal value [14].

Each bacterium represents the position of a candidate window region. Initially a selected rectangular window reference region is used for tracking within an image frame. Then the target object can be reached by a correlation matching between Hue color histograms level of the reference window image and the candidate sub region window of the image frame by collecting the hue values over image target pixels as it was shown in fig. 1 and fig. 2.

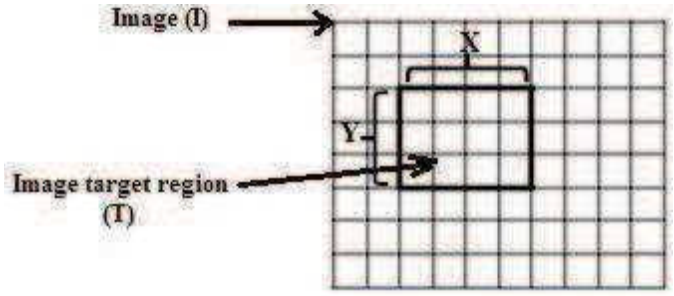

Figure 1. Example of a candidate sub region window of an image frame

Where X represents the width in pixels of T, Y represents the height in pixels of T, s is the number of hue level intervals in the histogram, H_i is the number of pixels of image target region T that figures in the i^{th} interval, H_i can take values between $[0, X \times Y]$, on the other hand, it is clear that $\sum H_i$ is equal to $X \times Y$.

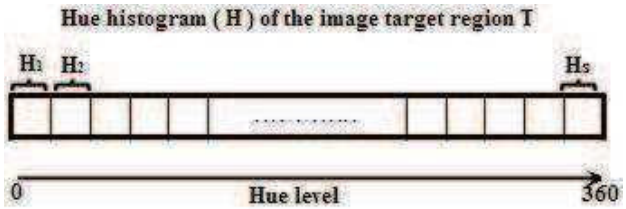

Figure 2. Hue level histogram

B. Fitness function

The fitness function f used in this paper is calculated as follows (function minimization):

Initially, a hue level color histogram $H_{Ref}(T_0)$ is extracted from the selected object region to track, and then it will be stored so far to be used like a reference, where T_0 is the selected image window reference.

To calculate the fitness of a candidate image region T_i during the optimization process, another histogram should be calculated for it (H_i). The similarity between the two histograms H_{Ref} and H_i is computed as the absolute value of the two histograms difference.

This similarity will have a value between 0 and $(2 \times X \times Y)$, 0 corresponds to full similarity and $(2 \times X \times Y)$ to any similarity.

The fitness function f is normalized to have a value between 0 and f_{max} (in our algorithms f_{max}=50) as follows:

$$f(T_i) = \frac{f_{max} \times \sum_{i=1}^{s} \text{abs}(H_i - H_{Ref})}{2 \times T_{i.x} \times T_{i.y}}. \quad (2)$$

IV. DYNAMIC AUTO-ADAPTIVE BACTERIAL FORAGING OPTIMIZATION (DABFO)

First, a dynamic BFO (DBFO) is presented, some modifications are introduced to the original BFO by reevaluating bacteria cost in the beginning of each elimination dispersal step and by reevaluating the best position already stored so far in the end of the elimination dispersal step (the fitness value in a same position can be changed in time in a dynamic environment). Second, in addition to these modifications, our DABFO algorithm employs an auto-adaptive elimination dispersal parameter which guarantees a good diversity maintaining balance and an efficient exploration/exploitation of the search space to deal with dynamic environment changes. DABFO also use an auto-adaptive prediction technique presented in the following paragraphs.

A. Auto-adaptive elimination dispersal

To maintain a flexible diversity of bacteria in the search space, the fixed probability parameter of elimination and dispersal P_{ed} is replaced by a dynamic one, its value is determined from the current best fitness and from the bounds of the best fitness in previous states (f_{c_max} and f_{c_min}), this probability parameter of elimination and dispersal is calculated to be between the maximum and the minimum probability ($P_{ed_{max}}$ and $P_{ed_{min}}$) in Equation (3) as:

$$P_{ed} = Coef \times \left(P_{ed\_max} - P_{ed\_min}\right) + P_{ed\_min}. \quad (3)$$

$$Coef = \frac{(f(Cell_{best}) - f_{min})}{(f_{c\_max} - f_{c\_min})}, \quad (4)$$

Where P_{ed_max} and P_{ed_min} are the maximum and the minimum probability of a cell being subjected to elimination and dispersal.

This auto-adaptive parameter P_{ed} plays an important role to control the diversity ratio; when the optimum is closely tracked (good fitness) P_{ed} is automatically decreased to favorite exploitation by reproduction and prediction, and when the optimum is farther (bad fitness) P_{ed} is increased to allow more diversity and discovering other regions in the search space.

B. Auto-adaptive prediction

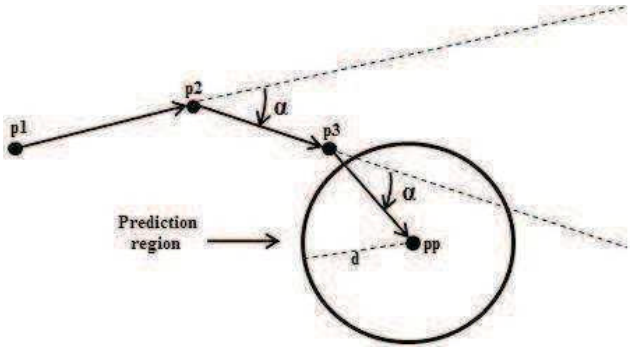

Figure 3. Auto-adaptive prediction

This technique ensures a rapid and efficient tracking based on three auto-adapted elements:

First, the position of the prediction region: based on trajectory and velocity change of the tracked optimum; From the last three memorized optimum positions p_1, p_2 and p_3, the next position p_p will be determined. An angle α should be calculated, then the distance of the vector $\overrightarrow{p_2p_3}$ is used with the angle α to form a new vector $\overrightarrow{p_3p_p}$ which predicts the new position p_p like it is shown in Fig. 3, in which p_p represents the position where the population prediction part will be randomly placed around.

Where $p_{p.x}$ and $p_{p.y}$ are defined by Equation (5) and Equation (6):

$$p_{p.x} = d_0 \times \cos(\beta) + p_{3.x}. \quad (5)$$

$$p_{p.y} = d_0 \times \sin(\beta) + p_{3.y}. \quad (6)$$

$$\beta = 2 * \operatorname{atan}\left(p_{3.y} - p_{2.y}, p_{3.x} - p_{2.x}\right) - \operatorname{atan}\left(p_{2.y} - p_{1.y}, p_{2.x} - p_{1.x}\right), \quad (7)$$

$$d_0 = \sqrt{(p_{3.x} - p_{2.x})^2 + \left(p_{3.y} - p_{2.y}\right)^2}, \quad (8)$$

Second, size of the prediction region: The prediction region is defined by a circle of center p_p and a variable diameter d (defined from the last optimum velocity), therefore, the prediction region size is automatically adapted to the optimum velocity using Equation (9).

$$d = C_p \times d_0\ , \quad (9)$$

Where C_p is a constant coefficient.

Third, Number of individuals intended for prediction: This number can be determined by the same idea proposed in A, and for the same reasons, it is calculated as:

$$p_{nbr} = Coef \times (P_{max} - P_{min}) + P_{min}. \quad (10)$$

Where P_{max} and P_{min} are the maximum and the minimum proportions of the population size intended for prediction.

C. Pseudo code for the DABFO algorithm

In addition to the original BFO parameters, DABFO use other parameters as follows: f_{max} and f_{min} are the maximum and the minimum fitness found so far in the current period, P_{ed_max} and P_{ed_min} are the maximum and the minimum probability of a cell being subjected to elimination and dispersal, P_{max} and P_{min} are the maximum and the minimum proportions of the population size intended for prediction, $Coef$ is a linear auto-adaptive coefficient, p_1, p_2 and p_3 are $Cell_{best}$ last three positions, p_p is the new predicted position, d is the half of the region diameter multiplied by a constant coefficient c_p, p_{nbr} is the number of cells intended for prediction.

Population ← InitializePopulation($Cells_{num}$, $Problem_{size}$);
for l = 0 to N_{ed} do
  Reevaluate(Population);
  for k = 0 to N_{re} do
    for j = 0 to N_c do
      ChemotaxisAndSwim(Population, $Problem_{size}$, $Cells_{num}$, N_s, $Step_{size}$, $d_{attract}$, $w_{attract}$, $h_{repellant}$, $w_{repellant}$);
      foreach *Cell* □ Population do
        if Cost(*Cell*) ≤ Cost($Cell_{best}$) then
          $Cell_{best}$ ← *Cell*;
        end
      end
    end
    SortByCellHealth(Population);
    Selected ←
    SelectByCellHealth(Population, $Cells_{num}$);
    Population ← Selected+ Selected;
  end
  Reevaluate($Cell_{best}$);
  f_{max}, f_{min} ← CalculateCostMaxMin($Cell_{best}$);
  $Coef$ ←CalculateCoefficient(f_{max}, f_{min}, Cost($Cell_{best}$));
  P_{ed} ← AdaptEliminationDispersal($Coef$, P_{ed_max}, P_{ed_min});
  foreach *Cell* ∈ Population do
    if Rand() ≤ P_{ed} then
      Cell ← CreateCellAtRandomLocation();
    end
  end
  p_1, p_2, p_3 ← SheeftAndMemorizeLastBest(p_1, $p_2, p_3, Cell_{best}$);
  p_p, d ← Prediction(p_1, p_2, p_3);
  p_{nbr} ← PredictionPartNumber($Coef$, P_{max}, P_{min});
  foreach *Cell* ∈ at low p_{nbr} of Population cells do
    Cell ← CreateCellAtPredictionRegion(p_p, d);
  end
end
return $Cell_{best}$;

V. Experiments And Results

A. Environment setting

The experiments are set in a testing environment of 3 videos (A, B and C) from the public BoBoT benchmark (available at http://www.iai.uni-bonn.de/~kleind/tracking/ for video-tracking algorithms and systems, which comprises several short video sequences showing arbitrary target objects. Experiments have been used for all frames of each video. Figs. 5, 6 and 7 show only the first 120 frames of each video. All videos are in 320x240 and 25fps format. The algorithms are implemented in MATLAB R2012a.

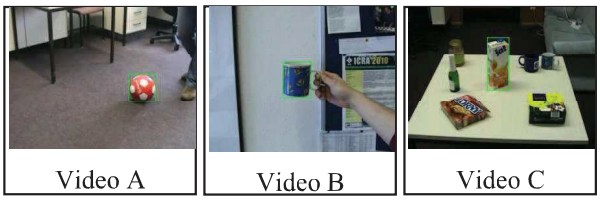

Figure 4. Sample video frames from the BoBoT benchmarks [8]

B. Video scrolling in the algorithms

For the identification of an object in a frame, a rectangle window is utilized to represent the position of a candidate bacterium. After each iteration of the elimination-dispersal, the best bacterium fitness determines the target object of the current frame. Each frame in a video is continuously treated by the algorithm without reinitializing the parameters.

C. Parameters Selection

In all experiments the following parameters are initialized as: $Cells_{num} = 20$, $N_{ed} = 1$, $N_{re} = 1$, $N_c = 4$, $N_s = 3$, $P_{ed} = 0.25$, $P_{ed\_max} = 0.4$, $P_{ed\_min} = 0.1$, $P_{max} = 0.6$, $P_{min} = 0.2$, $C_p = 1.5$.

D. Performance Metrics

The Accuracy Performance $Accuracy$ and the Collective Mean Fitness F_C measures are used (In our case f_{min}, f_{max} are known 0 and 50).

$$Accuracy_i = f\left(generation_{best_i}\right) - f_{min}\,. \qquad (11)$$

$$F_C = \frac{1}{N} \times \sum_{i=1}^{N} f\left(generation_{best_i}\right). \qquad (12)$$

E. Expiremental Results

All tests are averaged across 30 runs. Videos A, B and C have respectively sizes of 602, 629 and 404 frames. Figs. 5, 6 and 7 show the DBFO and DABFO graphs of the average accuracy performance of the first 120 frames in each video. The collective mean fitness in each video is recapitalized in Table I. Table II represents a comparative statistics of the number of frames better tracked by DABFO than those tracked by DBFO in each video.

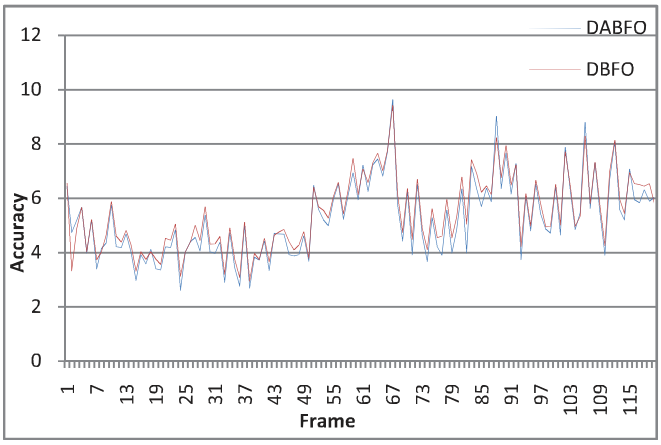

Figure 5. Average accuracy performance in Video A

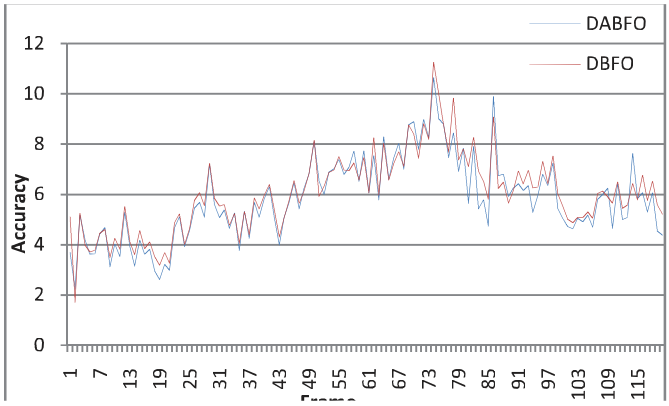

Figure 6. Average accuracy performance in Video B

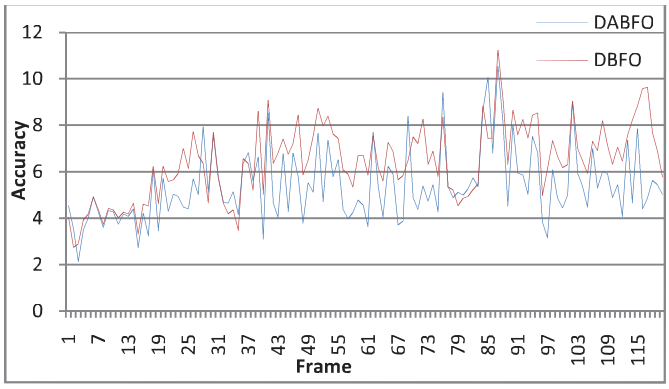

Figure 7. Average accuracy performance in Video C

TABLE I. AVERAGE COLLECTIVE MEAN FITNESS

Algorithm	Average collective mean fitness		
	Video A	*Video B*	*Video C*
DBFO	6.0350	10.1135	8.0972
DABFO	5.7932	9.8888	7.5305

TABLE II. AVERAGE NUMBER OF FRAMES BETTER TRACKED STATISTICS

Number of frames better tracked by DABFO than DBFO (frames number/total frames)		
Video A	*Video B*	*Video C*
527/602	500/629	291/404

F. Discussion of Results

In all experiments, both DBFO and DABFO algorithms can track the object in all video frames with a good precision. The stability form of DBFO and DABFO graphs and their similarity form mean that the object is continuously tracked as it is illustrated on Figs. 5, 6 and 7.

Despite that both DBFO and DABFO algorithms use the same parameter values (except the auto adaptive parameters) and the same number of execution iterations, precision of tracking in DABFO is better than DBFO as specified in Table I. Furthermore, when the changes severity and velocity are large and not stable like it is in video C and unlike videos A and B, results show that DABFO outperforms DBFO significantly (see Fig. 7), where DABFO will be rapidly auto-adapted to the environment changes and shows more performance. Table II shows that about 80% of video frames are better tracked using DABFO algorithm.

VI. CONCLUSION

In this paper we have presented a dynamic BFO algorithm version for object tracking problem by using Hue level histograms similarity method of the HSV color space. Also an auto-adaptive version of the dynamic BFO called DABFO is presented, in which several auto adaptive techniques have been introduced (the position of the prediction region, size of the prediction region, number of individuals intended for prediction, and probability parameter of elimination and dispersal) to enhance the exploration-exploitation trade-off and to unsure a flexible diversity maintaining. Our DABFO algorithm competes and even outperforms existing algorithms based on BFO especially when severity and velocity changes are large and not stable.

REFERENCES

[1] T. T. Nguyen, S. Yang, and J. Branke, "Evolutionary dynamic optimization: A survey of the state of the art," Swarm and Evolutionary Computation, Vol. 6, Oct 2012, pp. 1–24.

[2] C. Li, "Particle Swarm Optimization in Stationary and Dynamic Environments," Phd thesis, Leicester University, United Kingdom, December, 2010.

[3] W. J. Tang, Q. H. Wu, and J. R. Saunders, "Bacterial foraging algorithm for dynamic environments," IEEE Cong. Evol. Comp. July 16-21, 2006, Canada, pp. 1324-1330.

[4] T. T. Nguyen, B. Bhanu, "Real-Time Pedestrian Tracking with Bacterial Foraging Optimization," IEEE Ninth International Conference on Advanced Video and Signal-Based Surveillance, 2012.

[5] J. Brownlee, "Clever Algorithms: Nature-Inspired Programming Recipes," LuLu. ISBN: 978-1-4467-8506-5, January, 2011, pp 257-263.

[6] P. Gupta, K. Sharma, and A. P. Singh, "A Review of Object Tracking Using Particle Swarm Optimization," VSRD International Journal of Electrical, Electronics & Comm. Engg. Vol. 2 (7), 2012.

[7] P. Sebastian, Y. V. Voon, and R. Comley, "Colour Space Effect on Tracking in Video Surveillance," International Journal on Electrical Engineering and Informatics , Vol. 2, Number 4, 2010, pp.292–312.

[8] D. A. Klein, "BoBoT - Bonn Benchmark on Tracking," www.iai.uni-bonn.de/~kleind/tracking/

[9] K. M. Passino, "Biomimicry of Bacterial Foraging for Distributed Optimization and Control," IEEE Control Systems Magazine, Vol. 22, No. 3, June 2002, pp. 52-67.

[10] Y. Zheng and Y. Meng, "Adaptive Object Tracking using Particle Swarm Optimization" Proceedings of the 2007 IEEE International Symposium on Computational Intelligence in Robotics and Automation Jacksonville, FL, USA, June 20-23, 2007.

[11] C. C. Hsu and G. T. Dai, "Multiple Object Tracking using Particle Swarm Optimization" World Academy of Science, Engineering and Technology 68 2012.

[12] J. Vergés-Llahí, J. Aranda and A. Sanfeliu, "Object Tracking System using Colour Histograms" 9th Spanish Sym. Pattern Recog.Image Anal., Castellon, May 2001.

[13] M. Mason and Z. Duric, "Using Histograms to Detect and Track Objects in Color Video" AIPR, 2001, pp. 154–162.

[14] Pradeep, M. Namratha and G. V. Manu, "Global and Localized Histogram Equalization of an Image" IJCER Vol. 2 Issue. 6, October 2012, pp. 238–252.