République Algérienne Démocratique et Populaire

وزارة التعليم العالي و البحث العلمي

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

UNIVERSITE DES SCIENCES ET DE LA TECHNOLOGIE d'Oran Mohamed Boudiaf

Faculté des Sciences

Département d'Informatique

Spécialité : Informatique

THESE

Présentée par

Mr. Benyamina Ahmed

Pour l'obtention du diplôme de Doctorat en Sciences en Informatique

Thème

Application des algorithmes de colonies de fourmis pour l'optimisation et la classification des images

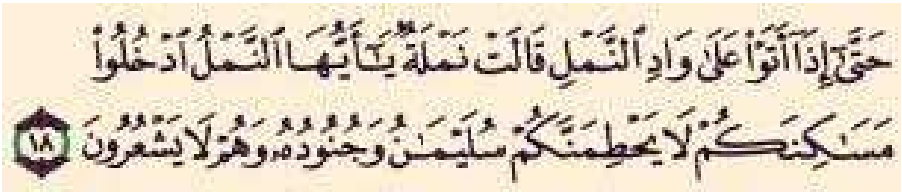
Soutenue le 08 avril 2013

Devant le jury composé de :

Président	Mr BENYETTOU Mohamed	Professeur	USTO(MB)
Rapporteur	Mme FIZAZI Hadria	Maître de conférences	USTO(MB)
Examinateur	Mr BENYETTOU Abdelkader	Professeur	USTO(MB)
Examinateur	Mr BELDJILALI Bouziane	Professeur	UO(Senia)
Examinateur	Mr KHALFI Fayçal	Professeur	UO(Senia)
Examinateur	Mr SIMOHAMED AREZKI Mohamed	Directeur de recherche	CTS(Oran)

Année universitaire : 2012-2013

Le Quotidien britannique « The Times » a publié un article le 6 février 2009, sous le titre (Hills are alive with the sound of ants - talking to each other) pour les fourmilières vivantes avec des voix, en parlant avec les uns des autres). L'article parle des nouvelles découvertes scientifiques sur le langage de communication et la conversation dans le royaume de fourmis. L'article mentionne que les découvertes récentes ont montré que la langue de communication chez les fourmis est sophistiquée et avancée significativement plus qu'on ne le pensait auparavant. En insérant des microphones et des enceintes miniatures dans les nids, les chercheurs ont découvert que la reine pouvait donner des instructions à ses ouvrières. Ils ont enregistré ses "discours", et ont également découvert que d'autres insectes imitent ces messages, pour faire des fourmis leurs esclaves. Ainsi, les chenilles d'une espèce de papillon détournent les fourmis de leur tâche pour se faire nourrir. Il a été indiqué dans le Saint Coran :



Quand ils arrivèrent à la Vallée des Fourmis, une fourmi dit: "O fourmis, entrez dans vos demeures, (de peur) que Salomon et ses armées ne vous cassent (écrasent) sans s'en rendre compte.

Si nous prenons la nature pour guide,
nous ne nous égarerons jamais.
[Cicéron]

Remerciements

Je tiens à exprimer ma profonde gratitude à Mme FIZAZI Hadria *Maître de conférences à l'Université des Sciences et de le Technologies d'Oran – USTO- pour avoir dirigé mes travaux de recherche depuis le magistère et pour la confiance et l'intérêt qu'elle m'a témoignés durant ces années de thèse. Par ses conseils son dévouement constant et ses nombreuses discussions il a permis à ce travail d'être ce qu'il est aujourd'hui. Je la remercie également pour la liberté et la confiance qu'elle m'a toujours accordée et pour la démarche scientifique rigoureuse et l'esprit d'auto-critique qu'elle a su m'inculquer. J'espère avoir été à la hauteur de ces espérances.*

J'adresse mes remerciements les plus sincères à : Mr BENYETTOU Mohamed *pour l'honneur qu'il me fait en acceptant de présider le jury de cette thèse.*

Mes vifs remerciements vont aux membres du jury qui ont accepté de prendre de leur temps pour examiner mon travail : Mr BENYETTOU Abdelkader, Mr BELDJILALI Bouziane, Mr KHALFI Fayçal et Mr SIMOHAMED AREZKI Mohamed.

Je tiens à exprimer ma reconnaissance aux Monsieurs Naceri Abdelfettah et Guesbaoui Brahim *. Maîtres de conférences sont tous des enseignants à l'université de Béchar, pour leurs conseilles et leurs aides.*

Je remercie mon épouse pour le soutien et ses encouragements pour la finalisation de ce travail.

Enfin, un grand MERCI *à ma famille pour le soutien et les encouragements qu'elle a su m'apporter pour arriver au terme de cette thèse. J'espère l'honorer avec ce travail.*

Dédicace

Merci mon dieu de m'avoir aidé et guidé.

Je dédie ce modeste travail à ceux que j'aime et respecte.

A celle qui partage avec moi ma vie.

A mon père, à ma mère, à mes sœurs à mes frères et à mes enfants Ayat, Youcef, Yahia et Ali, je vous aime tous.

Benyaminaa@yahoo.com

Table des matières

Résumé

Actuellement, le monde naturel est devenu l'outil de base dans les recherches scientifiques. Car l'invention passe par une imitation d'un phénomène physique, transformée à une formulation mathématique et devenue un modèle mathématique.

Ces dernières années, l'intelligence collective a attiré un grand intérêt la plupart des chercheurs à la fois en biologie et en informatique. Les biologistes qui s'intéressent aux sociétés d'insectes cherchent à comprendre les mécanismes qui régissent les processus complexes qui conduisent à des comportements collectifs (recherche de la nourriture pour la survie d'une colonie, construction de nids, division du travail, …). Les informaticiens s'inspirent des phénomènes étudiés pour développer de nouveaux algorithmes distribués et adaptatifs en résolution de problèmes (reconnaissance de formes, classification, optimisation, routage dans des réseaux,…).

Les problèmes de l'optimisation et de la classification sont la base de toutes opérations vitales et beaucoup de méthodes ont été élaborées pour résoudre les objectifs de l'optimisation de partitionnement.

Parmi ces méthodes, les algorithmes de colonies de fourmis qui forment une classe des méta-heuristiques récemment proposée pour ces types de problèmes.

On va appliqués ces algorithmes pour l'optimisation et la classification de plusieurs types de données (images satellites et la base Machine Learning Repository de l'UCI et autres) et de voir leur apport à la résolution de ces types de problèmes.

***Mots clés* :** Classification, optimisation, classification non supervisée, colonies de fourmis, Images satellites, Méta-heuristique, AntClust, AntClust adapté, ACOClust, Biomimétique, Machine Learning Repository.

Abstract

Currently, the natural world has become the basic tool in scientific research. Because the invention is an imitation of a physical phenomenon, transformed to a mathematical formulation and become a mathematical model.

In recent years, collective intelligence has attracted great interest most researchers in both biology and computer science.

Biologists interested in insect societies seek to understand the mechanisms underlying the complex processes that lead to collective behavior research (foraging for survival of a colony, nest building, division of labor, ...) .

It inspired the phenomena studied to develop new distributed and adaptive algorithms solving (pattern recognition, classification, optimization, routing in networks,...).

The problems of optimization and classification are the basis of all vital operations and many methods have been developed to address the objectives of optimization and partitioning.

Among these methods, ants colonies algorithms that form a class of metaheuristics recently proposed for these types of problems.

We will apply these algorithms for optimization and classification of several types of data (satellite images and the base of Machine Learning Repository UCI and others) and see their contribution to solving these types of problems.

keywords : Classification, optimization, clustering, ants colonies, Satellites images, Meta-heuristic, AntClust, AntClust adapted, ACOClust, Biomimetic, Machine Learning Repository.

Liste des figures

Liste des tableaux

Liste des algorithmes

Liste des abréviations

ACO	Ant Colony Optimization
AS	Ant System
ISODATA	Iterative Self-Organizing Data Analysis Technique yAy!
KV	Kilo Volt
MW	MégaWatt
OCF	Optimisation par les Colonies de Fourmis
PSO	Particle Swarm Optimization
PAPSO	Parrallel Asynchronous
p.u.	Unité réduite ou *per unit*
SIA	Systèmes Immunitaires Artificiels
SMA	Systèmes Multi Agents
TSP	Traveling Salesman Problem

Introduction générale

Introduction générale

Les études éthologistes ont montré que dans la nature, les petites créatures faibles que sont les fourmis, arrivent à résoudre collectivement des problèmes quotidiens nombreux et trop complexes pour une seule fourmi tels que : recherche de nourriture, construction du nid, division du travail et allocation des tâches entre les individus, avec une organisation excrément[1] structurée et sans aucune supervision

Par les comportements simples de chacune des fourmis, des interactions limitées à travers une coopération inconsciente, émergent des comportements collectifs intelligents et des modèles d'auto-organisation [Bonabeau, 2000]. Les fourmis sont devenues dès lors une nouvelle source d'inspiration pour la conception de méthodes de résolution de problèmes complexes. De plus cette source d'inspiration n'est pas unique étant donné que les fourmis sont dotées d'une grande diversité de caractéristiques disjointes et de comportements collectifs variés. Une nouvelle classe d'algorithmes est alors apparue sous le nom « algorithmes de fourmis artificielles ». Leur popularité est due d'une part à la facilité de mise en œuvre et d'autre part à la complexité des fonctions réalisables [Deneubourg, 1990][Colorni, 1992][Dorigo, 1996][Van De Vijver, 1997] [Bonabeau, 1999][Topin, 1999][Wilson, 1985]. Deux comportements collectifs ont été principalement étudiés chez les fourmis : l'optimisation de chemin et le tri des cadavres. Le premier comportement appelé aussi fourragement permettent aux fourmis de retrouver le plus court chemin entre leur nid et une source de nourriture grâce à un système de marquage de phéromones. Ce comportement naturel a été modélisé et transposé à la résolution de nombreux problèmes d'optimisation combinatoires sous le nom d'une nouvelle méta-heuristique « optimisation par les colonies de fourmis ou OCF (ACO pour Ant Colony Optimization) ». Le deuxième comportement collectif des fourmis concerne la capacité de certaines espèces de fourmis à organiser collectivement des cimetières composés de cadavres empilés les uns sur les autres. Là aussi, les chercheurs ont exploité ce comportement pour fournir des algorithmes de classification pour lequel l'informatique classique n'a pas donné de solution satisfaisante.

Le problème d'optimisation Combinatoire a longtemps privilégié les méthodes exactes au détriment des méthodes approchées. Les raisons de cette préférence sont multiples et tiennent tant à des facteurs historiques que théoriques. En particulier, le recours à des modèles exacts permet de bénéficier de résultats théoriques forts accompagnant les notions de convergence et d'optimalité globale. De fait, plusieurs problèmes classiques tels que le Voyageur de Commerce peut être résolu de manière exacte en un laps de temps raisonnable grâce à des techniques extrêmement performantes (par exemple le Branch and Cut).

Cependant, depuis la fin des années 80, les méthodes approchées, et plus particulièrement les méta-heuristiques suscitent un intérêt croissant de la part de la communauté. Ce succès tient pour une grande part à leur capacité à fournir des solutions d'excellente qualité au prix d'une consommation en ressources réduite. La perte du caractère optimal se voit donc compensée par la diminution des temps de calcul et donc par un accroissement de la capacité de réaction. Les méta-heuristiques bénéficient également d'autres avantages significatifs tels que la faculté à s'adapter rapidement à des modifications structurelles du problème (ajout ou suppression de contraintes).

[1] Les **excréments** sont toutes les matières naturellement évacuées par un organisme animal, sous forme solide ou liquide : matières fécales, urine, sueur, etc..

Ces caractéristiques les rendent parfaitement adaptées aux exigences du milieu industriel, ce qui explique l'arrivée sur le marché d'un nombre croissant d'outils d'aide à la décision intégrant des méta-heuristiques.

Parmi ces méta-heuristiques, nous pouvons citer les algorithmes génétiques, le recuit simulé, les réseaux de neurones, les algorithmes à estimation de distribution, l'optimisation par essaim de particules et l'optimisation par colonie de fourmis.

De l'autre côté, les méthodes de classification automatiques affectent chaque objet à une classe, en fonction d'un ou de plusieurs attributs de cet objet. La classification est dite supervisée lorsque des informations à priori sont utilisées pour la construction de classes sous la forme d'un ensemble d'apprentissage. Dans le cas où aucune connaissance à priori n'est disponible, on parle de classification non supervisée. Dans ces type de méthodes, on trouve les méthodes hiérarchiques constituées d'une suite de partitions emboîtées et les méthodes par partitionnement qui fournissent une seule partition.

De nombreux algorithmes de partitionnement déterministes existent dans la littérature telle que le K-means et le ISODATA (Iterative Self-Organizing Data Analysis Technique) et leurs variantes. Ces algorithmes sont très simples à implémenter et convergent rapidement avec une solution localement optimale.

Cependant leur majeur inconvénient est qu'ils nécessitent de fournir en entrée une partition initiale de bonne qualité ainsi que le nombre possible de classes. Ces contraintes rendent l'utilisation de ces algorithmes peu intéressante quand on veut classifier automatiquement un ensemble des données.

Dans ces vingt dernières années, une évolution croissante du nombre d'études menées sur les animaux vivant en groupe ou en société et plus particulièrement les insectes sociaux. Ces études éthologiques dans la théorie de l'auto-organisation ont inspiré un grand nombre de chercheurs pour développer des algorithmes de colonies de fourmis et essaims particulaires.

Les études menées sur les fourmis qui ont la particularité d'employer une substance chimique ou phéromone pour communiquer entre elles afin d'exécuter une tâche ou pour marquer leur trajet ont donné naissance aux algorithmes de colonies de fourmis artificielles et a fait l'objet de plusieurs travaux dans différents domaines d'application tels que la robotique, l'optimisation combinatoire basé sur le comportement de fourragement des fourmis ou encore la classification automatique qui s'inspire du comportement collectif d'agrégation/ségrégation des fourmis envers le couvain.

Les chercheurs reprennent les travaux existants pour la classification et utilisent les principes d'exploration stochastique et distribuée d'une population de fourmis artificielles pour fournir un partitionnement d'un ensemble des données en des classes pertinentes sans disposer d'une partition de départ et sans connaître le nombre de classes qui seront nécessaires. Selon le phénomène de stigmergie la partition optimale émerge à partir de l'activité collective de l'ensemble des fourmis et des interactions locales entre les fourmis et leur environnement.

L'intégration de telles aptitudes des fourmis ont poussé plusieurs chercheurs de différents domaines (biologie, optimisation,…) de rapprocher ces créatures vers la réalité est de résoudre les problèmes de tous les jours avec les techniques des fourmis. Ces techniques ont été traduites en algorithmes nommés algorithmes de colonies de fourmis.

Afin d'entamer ces algorithmes et de les faire rapprocher de la réalité, il est nécessaire de les appliquer de plus près et de juger leurs apports et leurs points forts.

Il est à noter que l'objectif initial du sujet est d'appliquer les algorithmes de colonies de fourmis pour les images mais il nous a été souhaitable de diagnostiquer les algorithmes de colonies de fourmis avec différents types de données et de sortir en dernier lieu avec une conclusion globale sur la possibilité ou non de l'application générale de ces algorithmes y compris les images.

Dans le cadre des travaux de cette thèse, on va donc chercher à traiter quelques algorithmes de colonie de fourmis pour résoudre le problème d'optimisation et de classification non supervisée sur ***différents types de données*** et de l'autre coté de bien voir leur points forts par rapport aux autres méthodes. La multitude des données utilisées a été pour objectif d'un coté de montrer la possibilité de manipulation de ces algorithmes sur n'importe qu'elle type de données et de l'autre coté on a trouvé des difficultés d'avoir des données concernant les images satellites !!!. Malgré tout cela, on va utiliser les images présentes dans le laboratoire de mes expériences au cours de la partie classification seulement.

Ensuite, on analyse les résultats obtenus et on passe à une phase avancée visant à améliorer ces techniques. Pour l'optimisation, on essaye de résoudre un problème réel et pour la classification une seconde opération qui sera effectuée par nos algorithmes proposés et on passe finalement à la comparaison entre ces algorithmes et l'évaluation des résultats obtenus.

L'évaluation des résultats d'algorithmes de classification, ainsi que la comparaison de tels algorithmes, reste encore aujourd'hui une problématique ouverte importante. La difficulté vient principalement du fait que de telles évaluations sont subjectives par nature car il existe souvent différentes manières pertinentes de regrouper un même ensemble de données [Candillier, 2006].

1. Motivations

Premièrement, à chaque problème d'optimisation on peut associer un problème de décision dont le but est de déterminer s'il existe une solution pour laquelle la fonction objectif soit supérieure (resp. inférieure) ou égale à une valeur donnée. En effet, la complexité d'un problème d'optimisation est liée à celle du problème de décision qui lui est associé. En particulier, si le problème de décision est ***NP-complet***, alors le problème d'optimisation est dit ***NP-difficile***.

Deuxièmement, le problème de classification revient à chercher une partition qui regroupe d'une manière optimale les N objets en K classes (clusters) de telle sorte que les entités d'une même classe soient plus proche entre eux en terme d'un (ou plusieurs) critère(s), qu'avec les objets des autres classes. On peut essayer de résoudre ce problème par une méthode brute en engendrant toutes les partitions possibles et à retenir celle qui minimise au mieux le critère de partitionnement. Malheureusement, la taille de l'espace des partitions possibles est de l'ordre de : $O\left(\frac{N^k}{k!}\right)$.

La classification se ramène alors à un problème d'optimisation complexe ***NP-difficile*** pour lequel les méthodes locales s'avèrent très vite impraticables même pour une image de petite taille.

Les méthodes classiques de résolution de problèmes consistent à décomposer le problème en sous problèmes et à définir dés le départ les étapes de résolution. Si auparavant cette méthode de résolution a donné de bons résultats, son efficacité a été remise en cause ces dernières années avec la complexité croissante des problèmes à résoudre et l'apparition de nouveaux besoins reflétant la nécessité de disposer de systèmes robustes et fiables dans des domaines dynamiques et incertains. Face à ces nouvelles difficultés, il est devenu nécessaire voire inévitable de chercher de nouvelles solutions en explorant de nouveaux paradigmes que ceux habituellement utilisés.

Depuis quelques années, les chercheurs informaticiens ont trouvé en le monde naturel, une source d'inspiration inépuisable pour la conception de nouveaux systèmes informatiques. Il s'agit de puiser dans les comportements des êtres naturels de nouvelles approches pour la résolution de problèmes difficiles. Le rôle de l'informaticien est d'observer et comprendre les mécanismes et processus qui régissent les comportements dits « intelligents » de ces individus pour la résolution des problèmes courants, puis extraire à partir de ces études des modèles

implantables sur des machines dont les résultats pourront être validés par rapport à ceux observés dans la nature.

En Biologie par exemple, de nombreux systèmes naturels composés d'individus autonomes exposent des aptitudes à effectuer des tâches qualifiées de complexes sans contrôle global. De plus, ils peuvent s'adapter à leur milieu soit pour y survivre, soit pour améliorer le fonctionnement du collectif. C'est le cas des colonies d'insectes sociaux [Camazine, 2002] tels que les termites, fourmis [Bonabeau, 1997], ou araignées [Bourjot, 1999] qui font effectivement preuve de remarquables capacités pour effectuer des tâches telles que : la construction de nids complexes, la construction de pont, la recherche efficace de ressources, la capture de proies…

L'étude des déplacements collectifs de vols d'oiseaux migrateurs ou de bancs de poissons montre également le fait que la tâche collective est le résultat des interactions des individus autonomes [Theraulaz, 1997]. Le fonctionnement du système immunitaire est lui aussi représentatif du fonctionnement d'un système complexe composé d'un ensemble d'agents autonomes.

Tous ces systèmes naturels présentent un point commun : l'émergence d'un comportement global collectif et complexe à partir de simples interactions entre des insectes simples dotés d'une intelligence très réduite et ne possédant qu'une vision très partielle de leur environnement [Bonabeau, 1999]. Ce comportement émergeant leur permet de résoudre collectivement des problèmes très complexes Il paraît donc légitime et inévitable d'étudier ce phénomène d'émergence afin de pouvoir en comprendre le fonctionnement et être capable de l'utiliser comme nouvelle approche pour la conception de systèmes artificiels. On parle alors d'intelligence artificielle en essaim pour désigner une telle approche.

Parmi ces systèmes, on distingue les méthodes d'optimisation par essaim particulaires inspirés de l'étude de l'organisation de groupes d'animaux, [Eberhart, 2001], les algorithmes inspirés des essaims d'insectes volants [Aupetit, 2003], et les systèmes de fourmis artificielles largement utilisés ces dernières années pour résoudre des problèmes de classification ou d'optimisation [Dorigo, 1991]. Il s'agit d'une nouvelle approche qui s'intéresse aux comportements individuels des fourmis réelles, aux interactions entre ces entités autonomes et à l'émergence au niveau supérieur de l'ensemble du système de comportements complexes pouvant être qualifié d'intelligent.

Le problème de classification étant par sa nature un problème d'optimisation complexe difficile à résoudre, nous nous proposons dans ce travail de tirer dans cette étude des comportements des fourmis de nouvelles méthodes de classification flexibles et fortement distribuées qui permettent la formation des primitives d'une manière coopération, collective et guidée.

2. Contribution :

Dans le cadre de notre thèse, nous avons proposer de nouvelles solutions, d'un coté l'application efficace des algorithmes de colonies de fourmis pour l'optimisation dans un domaine réel, et de l'autre coté, l'adaptation des algorithmes de colonies de fourmis pour la classification.

Il existe en optimisation plusieurs types d'algorithmes regroupés en un seul nommé ACO. Malgré tout cela, on va essayer d'appliquer un des anciens algorithmes et de le comparer avec ce dernier et de sortir en dernier lieu sur la réalité d'unification.

Pour la classification, il existe deux approches à base fourmi.

La première approche modélise la faculté des fourmis à trier collectivement leur couvain ou à construire des cimetières. Ce modèle est relativement simple et est très stimulant pour résoudre le problème de classification[Deneub, 1990][Monm, 2009a][Monm, 2009b][Monm, 2000] [Ouadfel, 2006].

La deuxième approche consiste à s'inspirer du comportement de fourragement des fourmis et leur capacité à s'auto-organiser pour trouver le plus court chemin entre le nid et la source de nourriture. En effet via les concepts de mémoire collective représentée par la phéromone, de diversification et d'intensification, les fourmis arrivent à trouver une solution optimale sans aucune supervision [Dorigo, 1991]. Ces algorithmes ont été regroupés sous le nom générique de «Optimisation par Colonies de Fourmis» (OCF) ou en anglais « Ant Colony Optimization » (ACO).

On va appliqué ces algorithmes aux différents types de données (images satellites et bases de la Machine Learning Repository,…) et ensuite on a analysé leurs résultats et proposition de nouvelles techniques d'adaptation et d'optimisation afin d'améliorer leur rendements et de donner un résultat performant.

3. Structure de la thèse

Cette thèse est divisée en cinq chapitres :

Le premier chapitre expose quelques concepts théoriques exploités dans le travail de notre thèse qui ont conduit à l'apparition de plusieurs voies telles que l'optimisation et la classification.

Le deuxième chapitre est consacré aux différents travaux inspirés par les comportements collectifs des fourmis. Après une introduction générale sur les fourmis réelles, le chapitre se focalise sur les fourmis artificielles ainsi sur les algorithmes orientés optimisation et plus précisément sur la méta-heuristique d'ACO.

Le troisième chapitre traite sur la notion d'optimisation. En premier lieu, il présente la notion d'optimisation et ses différentes catégories et se focalise en dernier lieu sur les algorithmes de colonies de fourmis pour l'optimisation. En annexe, une présentation d'une liste de méthodes d'optimisation qui seront utilisées pour une étude comparative.

Le quatrième chapitre présente un tour d'horizon sur la classification. Par suite le chapitre se focalise sur les algorithmes de colonies de fourmis orientés classification.

Le chapitre 5 porte sur l'application des différents algorithmes de colonies de fourmis pour l'optimisation et la classification aux différents types de données choisis. Une étude comparative avec des algorithmes classiques, récente, heuristiques ou méta-heuristiques est prévue et sera suivie pour la classification d'une évaluation des résultats obtenus. Ce chapitre présente, dans des cas spécial, une adaptation de quelques algorithmes de fourmis testés.

A la fin, une conclusion générale sur l'ensemble de nos travaux ainsi qu'aux améliorations qui pourrait être apportées et aux perspectives qu'elle offre.

✓ **Séminaires :**

a- Avant phase doctorale

- Film vidéo : Représentant du CUB au CIAJ de Béchar. Premier salon national du film vidéo pour amateur. 1 Mai 1997
- Présentation des outils de recherche d'informations en Internet. Centre Universitaire de Béchar. Internet et didactique. 16-17 Avril 2000
- Moteur de recherche et la sémantique. Centre Universitaire de Béchar. Deuxième journée sur l'internet et la sémiologie. 16-17 Avril 2001
- Systèmes auteurs. Centre Universitaire de Béchar. Journées d'études sur les nouvelles technologies de l'information (internet et didactique). Du 16 au 17 Avril 2002.
- Télédétection, Images satelitaires et classification. Centre Universitaire de Béchar. Jetic'2007. 21-22 Avril 2007
- Conception d'une approche hybride pour une classification multi source des données de télédétection. Université de Béchar. CITIC'2008. 25-26 Octobre 2008

b- Durant phase doctorale

1. « *AntClust adapté pour un partitionnement des images satellitaires par colonies de fourmis* ». Université de Ouargla. COSI'2010. 18-20 Avril 2010
2. « *Efficacité de l'algorithme hybride ACOClust pour le partitionnement des images Satellitaires* ». Guelma Universiy. International Conference on Signal, Image, Vision and their Applications. SIVA'11. November 21-24, 2011 Guelma Algeria

✓ **Publication associée** :

« *The efficiency of the adapted AntClust algorithm for satellite images clustering* ».
Malaysian Journal of Computer Science (MJCS) Décembre 2011

✓ **Encadrements des étudiants (Ingénieurs et Master)**

1. Classification automatique des images satellitaires par les algorithmes de Colonies de Fourmis (AntClust) : 2008 – 2009
2. Application de méta-heuristique biomimétique d'optimisation pour la classification des images satellitaires (ACOClust): (Algorithmes de Colonies de Fourmis) : 2009-2010
3. Algorithme d'optimisation basée sur la classification de régions des images médicales : 2009-2010
4. Partitionnement de corpus textuel avec l'algorithme de colonies de fourmis ACOTextClust : 2010-2011
5. Evaluation des résultats de la méthode de la méthode de classification AntClust : 2010-2011
6. Evaluation de la méthode de classification ACOClust (Ant Colony Optimization Clustering) : 2011-2012
7. Etude comparative entre les algorithmes AntClust & AcoClust :2011-2012

Chapitre 1.

Concepts théoriques exploités dans notre projet

Chapitre 1 : Concepts théoriques exploités dans notre projet

1.1 Introduction

On était confronté durant la période de notre projet de thèse à plusieurs notions importantes, qui ont construit à travers les deux dernières décennies, une nouvelle génération d'algorithmes nommés « algorithmes de colonies de fourmis ».

J'ai commencé par les études éthologistes qui ont montré que dans la nature, les petites créatures faibles arrivent à résoudre collectivement des problèmes quotidiens nombreux et trop complexes. Par les comportements simples de chacune des insectes sociaux, des interactions limitées à travers une coopération inconsciente, émergent des comportements collectifs intelligents et des modèles d'auto-organisation [Bonabeau, 2000].

Entre fourmi biologique, l'insecte de base de notre travail, et du premier pas de l'inspiration biologique : la biomimétisme, les techniques d'émergence, de phénomène collectif des insectes sociaux, les méta-heuristiques qui sont les nouvelles techniques remplaçant les anciens qui ont données des résultats satisfaisants aux problèmes de reconnaissances de formes, arrivant aux autres notions qui sont la stigmérgie, le contrôle décentralisé , l'hétérarchie dense et autres notions importantes qui ne sont pas citées dans ce chapitre le seront présentées succinctement dans le glossaire en fin de ce mémoire.

1.2 La biomimétique

1.2.1 La biomimétique et l'informatique [Monm, 2000a]

Lorsqu'un nouveau problème se pose en informatique, et plus habituellement en ingénierie, il faut parfois définir de nouvelles méthodes de résolution car les techniques existantes ne sont pas précisément adaptées au cas traité. Ainsi, lorsque l'on veut inventer une nouvelle méthode de résolution de problème, il faut généralement une source d'inspiration. Celle-ci peut être totalement imaginaire et n'est pas obligée de faire référence au monde réel comme par exemple des méthodes mathématiques abstraites ou peut au contraire être issue de la modélisation des systèmes complexes naturels.

Il s'agit dans ce dernier cas de copier et d'adapter les concepts mis en œuvre par le monde du vivant pour la résolution de problèmes.

La source d'inspiration que constitue la biologie a de plus en plus de succès dans une branche de l'intelligence artificielle que l'on peut nommé informatique biomimétique.

Le principe de base de cette méthode de développement suit presque invariablement le même schéma et les mêmes motivations comme le fait remarquer Arkin [Arkin,1998] pour le domaine de la robotique biomimétique [Fg. 1.1].

Ce schéma s'applique à tout autre domaine, comme par exemple la mécanique ou l'informatique. Ce dernier domaine sera précisément le cadre principal dans lequel se place ce travail de thèse.

L'objectif du processus de création est de concevoir un modèle résolvant un problème ou une catégorie de problèmes en s'inspirant de schémas de comportements mis au point en éthologie. La première étape se base sur des études menées en biologie et consiste à extraire de ces études un modèle réalisable du point de vue informatique.

Les résultats obtenus par des simulations permettent ensuite d'évaluer la qualité du modèle relativement aux études biologiques. Les conséquences peuvent être utiles aux informaticiens afin d'améliorer le modèle lui même ou aux éthologistes qui peuvent développer et tester leur théories du comportement animal. Le double intérêt des deux disciplines n'est pas toujours immédiat et il faut reconnaître que c'est le plus souvent les informaticiens qui profitent d'abord de ces développements.

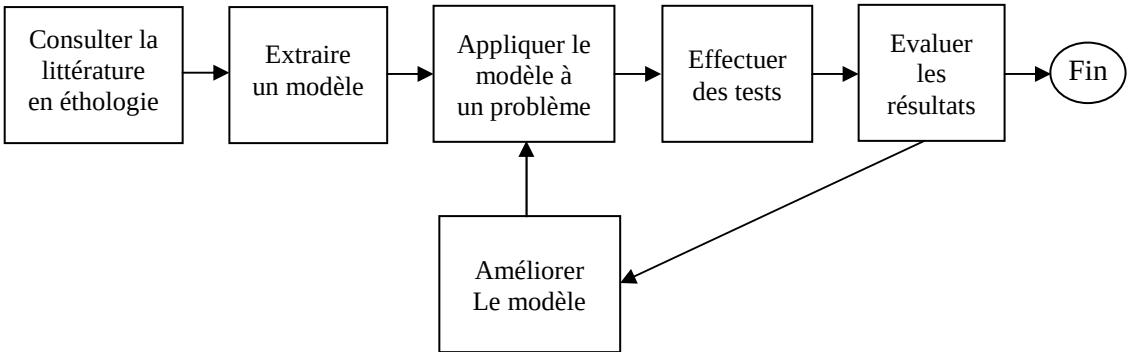


Fig. 1.1 Méthodologie de conception en informatique/robotique biomimétique (adapté de [Arkin, 1998)].

1.2.2 Les méthodes biomimétiques

Le premier travail sur la biomimétique était les algorithmes génétiques (AG).

Par suite, d'autres approches biomimétiques ont apparus. On peut citer comme exemple : Algorithmes multi-agents, Systèmes immunitaires et les algorithmes de colonies de fourmis et bien d'autres.

1.2.2.1 Algorithmes génétiques [Ouadfel, 2006]

Le problème de partitionnement étant un problème d'optimisation combinatoire.

Leurs sources d'inspiration est la faculté des êtres vivants à s'adapter aux contraintes de leur environnement via des évolutions génétiques.

Les premiers travaux utilisant les AG pour résoudre le problème de classification sont dus à [Raghavan, 1979]. Son algorithme, classifie une image, démarre avec un nombre de classes K qui est fixé à l'avance et une population de chromosomes de longueur N (taille de l'image à segmenter). Chaque chromosome associe une classe à un pixel. Les opérateurs génétiques, utilisés pour générer à chaque génération une population de partitions possibles, sont une adaptation des opérateurs génétiques binaires. Le but de l'algorithme est de minimiser une fonction fitness représentant l'inverse de la variance intra-classe.

Plusieurs variantes de cet algorithme de base ont été introduites par d'autres auteurs dans la littérature. Bhandarkar et Zhang [Bhandarkar, 1999] utilisent l'algorithme génétique pour minimiser une fonction fitness qui sera utilisée pour évaluer le résultat de la segmentation. La population initiale est générée par un processus aléatoire. La représentation de chaque chromosome contient un tableau d'étiquette, une information contour et un graphe d'adjacence. Un crossover de deux points est appliqué sur le tableau d'étiquettes et une mutation sur les contours de pixels sont employés. Dans [Maulik, 2003] un partionnement flou est réalisé en utilisant un algorithme génétique de codage réel. Une classification automatique sans connaissance a priori du nombre de classes présentes dans l'image est obtenue dans [Murthy, 1996] ,[Tseng, 2001],[Bandyopadhyay, 2002]. Dans [Andrey, 1998], un algorithme génétique distribué est utilisé pour segmenter des images texturées dans le cadre des champs de Markov. Le système de segmentation repose sur un principe évolutionnaire tel que la segmentation optimale émerge progressivement des interactions mutuelles entre les chromosomes de la population sans aucune information fournit a priori. Le système d'apprentissage génétique proposé par [Bhanu, 1995] permet d'adapter le processus de segmentation en fonction du type d'image à segmenter et des caractéristiques des objets à extraire.

1.2.2.2 Algorithmes multi-agents

> Ces algorithmes mettent en jeu des agents autonomes qui vont interagir directement ou indirectement entre eux ainsi que sur leur environnement pour résoudre le problème posé. [Labroche, 2003]

Les systèmes Multi-agents (SMA) sont nés au début des années 80 et se sont développés à partir de schémas de raisonnement ou d'organisations empruntées aux domaines de la vie et de la société [Ouadfel, 2006].

Dans les méthodes multi-agents, les agents en question n'ont qu'une vision locale du problème à traiter et interagissent entre eux ou avec leur environnement pour proposer une solution. Nous parlons alors d'Intelligence en essaim car la solution du problème émerge du travail collectif des agents.

Une des sources d'inspiration d'algorithmes montrant une intelligence en essaim est le déplacement des animaux en groupe ou aucun individu ne contrôle les autres mais pourtant des formes et des comportements complexes peuvent apparaître lors de ces déplacements.

> A titre d'exemple, les algorithmes PSO ("Particle Swarm Optimization") utilisent un ensemble de particules caractérisées par leur position et leur vitesse pour maximiser une fonction dans un espace de recherche. Des interactions ont lieu entre les particules afin d'obtenir des comportements globaux efficaces. [Azzag, 2004]

Ce principe a était appliqué en 1998 dans [Proctor, 1998] pour la première fois au problème de la classification automatique, où chaque agent représente une donnée

se déplaçant vers les données qui lui sont le plus similaires, permettant ainsi de créer des groupes de données similaires assimilables au classes de la partition.

Comme exemple des systèmes Multi-agents (SMA) à la classification des images, Bellet [Bellet, 1998] propose un système multi-agents spécialisé pour la classification d'images et L. Germond propose une méthode de segmentation coopérative pour la segmentation des images IRM [Germond, 1999][Ouadfel, 2006].

1.2.2.3 Systèmes immunitaires

> Les Systèmes immunitaires Artificiels (SIA) sont une modélisation du système immunitaire humain et animal, représentant généralement une population d'anticorps qui va avoir pour mission de reconnaître des antigènes qui n'appartiennent pas à l'organisme. [Labroche, 2003]

L'application des SIA à la classification automatique se fait en associant les données aux antigènes que devra détecter le système, et en les lui présentant de manière itérative jusqu'à atteindre un critère d'arrêt.

Ainsi les anticorps seront sécrétés en effectuant des mutations sur les anticorps suffisamment proches des antigènes rencontrés et en appliquant une sélection en diminuant le nombre d'anticorps se ressemblant.

A la fin des itérations des groupes d'anticorps serons obtenus, chaque groupe reconnaissant un type d'antigènes (de données). Ces groupes représenteront les classes obtenues au partitionnement.

Une description des fondements théoriques et de nombreuses applications des systèmes immunitaires artificiels peuvent être trouvées dans [De Castro, 1999], [De Castro, 2000] et dans [Dasgupta, 1997], mais aussi dans un livre de référence [Dasgupta, 1999] [Dréo, 2003].

1.2.2.4 Algorithmes de colonies de fourmis artificielles

Le comportement des fourmis réelles représente une grande source d'inspiration dans le domaine informatique. L'intérêt accordé aux fourmis a conduit à l'élaboration de quantités d'algorithmes basés sur des populations de fourmis artificielles et ceci notamment dans les domaines de la classification et de l'optimisation.

L'un des modèles les plus connus ACO (pour Ant Colony Optimization) a été introduit par [Colorni, 1991] initialement dans le cadre du problème du voyageur de commerce. Les fourmis utilisent des phéromones pour marquer des arcs entre les villes. Ces phéromones représentent en fait une distribution de probabilités qui est mise à jour en fonction des résultats observés (longueur totale du chemin par exemple). [Azzag, 2004]

Les algorithmes basés sur les populations de fourmis s'avèrent être des systèmes multi-agents dans le sens où la partition obtenue, dans le cadre d'un problème de partitionnement, émerge de l'activité conjointe des fourmis qui peuvent être assimilées aux agents d'un système multi-agent.

Pour la plupart des méthodes, la communication entre les fourmis se fait selon le principe de la stigmergie c'est-à-dire de façon indirecte grâce a des interactions entre les fourmis et leur environnement.

1.3 L'émergence en informatique et comportements collectifs des insectes sociaux

Bien que l'émergence soit encore aujourd'hui l'une des notions les plus floues et les plus discutées, elle est actuellement la plus utilisée pour la conception des systèmes artificiels. Son origine viendrait d'après Ali et Zimmer [Ali, 1997], La notion d'émergence peut être définie comme une propriété macroscopique d'un système qui ne peut pas être déduite à partir de son fonctionnement microscopique. [Ouadfel, 2006]

Les principales propriétés de l'émergence: [Grumbach, 1997]

- La nouveauté: un phénomène émergent et en quelque sort unique.
- La temporalité: le phénomène se construit dans le temps, il ne peut être instantané.
- La stabilité: le phénomène doit laisser le système dans lequel il se trouve stable.
- L'observation d'un phénomène apparent: en termes d'irréductibilité il est impossible de déduire à partir des propriétés de micro-niveau celles de macro-niveau
- La cohérence et la corrélation du phénomène: Le système doit être capable de modifier de son comportement en fonction de son environnement sans l'aide d'un superviseur
- L'observation d'une dynamique le phénomène n'est pas connue au départ, il s'auto-crée et s'auto-maintient.
- La non-linéarité : Ceci signifie que tout composant du système peut être influencé de manière indirecte par d'autres composants du système (qui ne lui sont pas directement liés).
- L'auto-organisation : Un système émergent est un système capable de s'auto-organiser et de changer de comportement en fonction de son environnement.

1.3.1 Les mécanismes de l'auto-organisation:

C'est un mécanisme connu par les biologistes [Camazine, 2001], peut être alors définie comme un moyen permettant à un système de se structurer et de se maintenir sans aucune intervention de l'extérieur.

Plusieurs définitions du concept d'auto-organisation existent dans la littérature.

Nous pouvons en citer les suivantes :

Définition1: « L'auto-organisation est une description d'un comportement, elle a une valeur heuristique et elle permet d'indiquer un phénomène. Elle est condamnée à rester une simple description, tant qu'on ne se préoccupe pas de rechercher le mécanisme qui est à son origine» [Varela, 1988]

Définition 2 : « Un système auto-organisateur est un système qui change sa structure de base en fonction de son expérience et de son environnement. » [Ünsal,1993]

Définition 3 : «L'auto-organisation est un processus où l'organisation (contrainte, redondante) d'un système croît de manière spontanée, i.e. sans que cet accroissement soit contrôlé par l'environnement ou ce qui l'entoure ou encore un système externe. » [Krippendorff, 1997].

Toutes ses définitions font référence à ces mêmes concepts : structuration, organisation, interaction, autonomie et enfin émergence d'un comportement global à partir de plusieurs comportements locaux [Georgé, 2004]. L'auto-organisation peut être alors définie comme un moyen permettant à un système de se structurer et de se maintenir sans aucune intervention de l'extérieur. Chaque composant du système réagit aux stimulus par des règles locales simples et modifie ainsi son environnement et donc le comportement des autres composants [Ouadfel, 2006],

La question cruciale est donc de comprendre comment les composants d'un système interagissent entre eux pour produire un modèle complexe (au sens relatif du terme, i.e. plus complexe que les composants eux-mêmes). Un certain nombre de phénomènes nécessaires ont été identifiés : ce sont les processus de *rétroaction* et la gestion des *flux d'informations* [Dubuisson, 2006].

La rétroaction (ou feed-back) qui est obtenue lorsque les résultats (obtenus par transformation des données d'entrée) sont retransmis au système sous la forme de nouvelles données d'entrée. Lorsque les nouvelles données agissent dans le même sens que l'action principale, c'est-à-dire amplifier la transformation, la rétraction est dit positive. Dans ce cas, ces effets sont cumulatifs. Dans le cas contraire la rétroaction est dite négative, et le système doit être stable.

Mais le système a besoin de composantes (des individus) pour assurer le renforcement des solutions précédentes, c'est l'exploitation de l'espace de recherche; et de composantes qui évoluent, volontairement ou pas, d'une façon aléatoire, ces dernières serviront au processus d'exploration de l'espace de recherche. En général, les composantes d'un système social peuvent assurer les deux processus (exploration / exploitation) [Tfaili, 2007].

Certaines espèces d'oiseaux vivent en colonies. Le fait de vivre en colonie a des bénéfices, comme une meilleure détection des prédateurs, ou encore la facilité de recherche des sources de nourriture. Dans ce cas, le mécanisme est l'imitation : des oiseaux d'une même espèce, qui cherchent à nicher, sont attirés par des sites où d'autres oiseaux de cette espèce nichent déjà, le comportement est le suivant « nicher à côté du lieu où l'autre a niché ». Chez les insectes sociaux, la rétroaction positive est illustrée par le renforcement des traces de phéromone, ce qui permet à la colonie toute entière d'exploiter des solutions anciennes ou récentes [Tfaili, 2007]. De même pour les bancs de poissons suivent le comportement "je te suis où tu vas".

Cependant, il y a un risque si la rétroaction positive agit toute seule, sans la présence d'une rétroaction négative.

Le fait que la rétroaction positive par nature s'amplifie peut engendrer un potentiel destructif ou explosif dans le système où elle opère. Le comportement est donc plus compliqué qu'il n'y paraît. Par exemple, pour un banc de poissons, le vrai comportement est le suivant "je vais où les autres vont, sauf s'il n'y a plus de place". Dans ce dernier cas, les rétroactions positive et négative peuvent être implémentées dans des règles comportementales [Tfaili, 2007].

Donc il est aisé de comprendre que les interactions entre les composants d'un système vont très souvent mettre en jeu des processus de communication, de transfert d'informations entre individus.

D'une manière générale, les individus peuvent communiquer, soit par le biais de signaux, c'est-à-dire en utilisant un moyen spécifique pour porter une information, soit par le biais d'indices, où l'information est portée accidentellement. De même, l'information peut provenir directement d'autres individus, ou bien passer par biais de l'état d'un travail en cours. Cette deuxième possibilité d'échanger des informations, par le biais de modifications de l'environnement, se nomme la *Stigmergie* [Dubuisson, 2006].

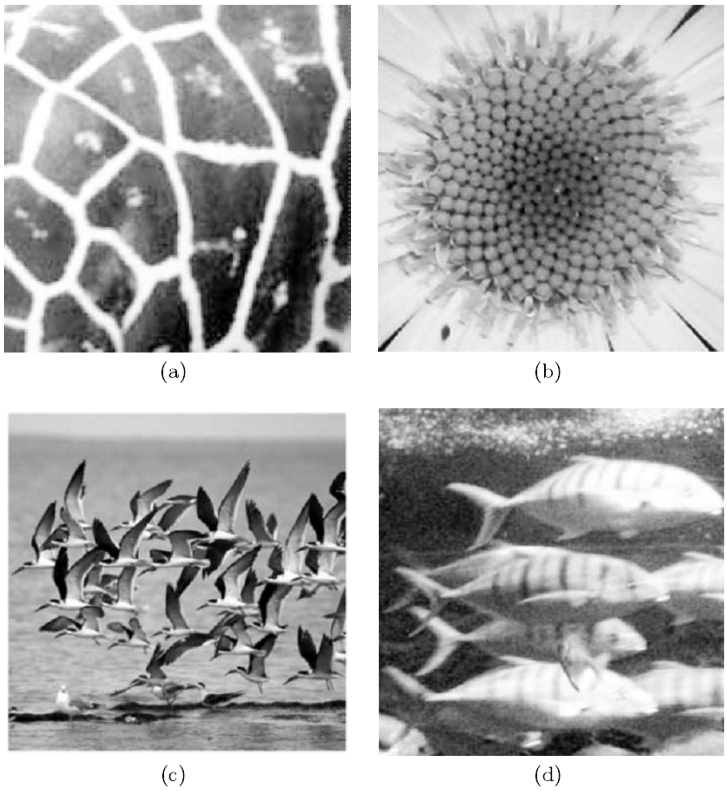


Fig. 1.2 Exemples de modèles observables dans des systèmes biologiques.

(a) motifs de la robe d'une girafe réticulée (U.S. Fis hand Wildlife Service, Gary M. Stolz), **(b)** double spirale de Fibonacci au cœur d'une pâquerette, **(c)** groupe d'oiseaux en vol, **(d)** poissons assemblés en banc.

1.3.2 Stigmergie:

Elle est précisément définie comme une "forme de communication passant par le biais de modifications de l'environnement", mais on peut rencontrer le terme "interactions sociales indirectes" pour décrire le même phénomène.les biologistes différentient la stigmergie "quantitative" de celle "qualitative", mais le processus en lui-même est identique. La grande force de la stigmergie est que les individus échangent des informations par le biais du travail en cours, de l'état d'avancement de la tâche globale a accomplir [Camazine, 2001]

Stigmergie = stigma (piqûre) **+ ergon** (travail ou œuvre) → simulation par travail

1.3.3 Contrôle décentralisé :

Dans un système auto-organisé, il n'y a pas de prise de décision à un niveau donné suivie d'ordres et d'actions prédéterminées. En effet, dans un système décentralisé chaque individu dispose d'une vision locale de son environnement, et ne connaît donc pas le problème dans son ensemble. Bien que, d'une manière générale, cette discipline tende à utiliser des modèles de comportements plus, complexes, fondés notamment sur les sciences de la cognition. Les avantages d'un contrôle décentralisé sont notamment la robustesse et la flexibilité [Bonab, 1999]. Systèmes robustes, car capables de continuer à fonctionner en cas de panne d'une de leurs parties; flexible car efficaces sur des problèmes dynamiques.

1.3.4 Hétérarchie dense :

L'hétérarchie dense est un concept issu directement de la biologie [Wilson, 1988] utilisé pour décrire l'organisation des insectes sociaux, et plus particulièrement des colonies de fourmis. Le concept d'hétérarchie décrit un système où les propriétés des niveaux globaux agissent plus ou moins sur les propriétés des niveaux locaux, mais également un système ou une activité dans les unités locales influence en retour les niveaux globaux. L'hétérarchie est dite dense dans le sens où un tel système forme un réseau hautement connecté, où chaque individu peut échanger des informations avec n'importe quel autre. Ce concept est en quelque sorte opposé à celui de hiérarchie où, dans une vision populaire mais erronée ; la reine gouvernerait ses sujets en faisant passer des ordres dans une structure verticale, alors que, dans une hétérarchie, la structure est plutôt horizontale (fig.1.3).

Fig. 1.3 Hiérarchie (a) et heterarchie dense (b) : deux concepts opposés

On constate que ce concept recoupe celui de contrôle décentralise, mais aussi celui de stigmergie, en ce sens que l'hétérarchie décrit la manière dont le flux d'information parcourt le système. Cependant, dans une hétérarchie dense, tout type de communication doit être pris en compte, tant la stigmergie que les échanges directs entre individus.

1.4 Méta-heuristique

1.4.1 Heuristique :

Les heuristiques sont des règles empiriques simples qui, à la différence des algorithmes, ne se basent non pas sur des analyses scientifiques parfois trop complexes (car nécessitant la définition et l'assignation de nombreux éléments), mais

sur l'expérience et les relations accumulées au fils des résultats. Plus simplement ces règles utilisent les résultats passés et leurs analogies afin d'optimiser leurs recherches futures en examinant d'abord les cas les plus plausibles. [Kévin, 2006]

1.4.2 Méta-heuristiques :

Parmi les heuristiques, certaines sont adaptables à un grand nombre de problèmes différents sans changements majeurs dans l'algorithme, on parle alors de méta-heuristiques[1] [Wiki, 2012a].

Les méta-heuristiques sont apparues dans les années 1980 et forment une famille d'algorithmes d'optimisation visant à résoudre des problèmes d'optimisation difficile pour lesquels on ne connaît pas de méthode classique plus efficace.[Wiki, 2012a]

Les méta-heuristiques sont généralement des algorithmes stochastiques itératifs, qui progressent vers un optimum global, c'est-à-dire l'extremum global d'une fonction, par échantillonnage d'une fonction objectif (voir chapitre 3 : problème d'optimisation). Elles se comportent comme des algorithmes de recherche, tentant d'apprendre les caractéristiques d'un problème afin d'en trouver une approximation de la meilleure solution (d'une manière proche des algorithmes d'approximation) (voir la figure Fig. 1.4). Elles sont souvent inspirées par des systèmes naturels, qu'ils soient pris en physique, en biologie de l'évolution ou encore en éthologie. [Wiki, 2012a]

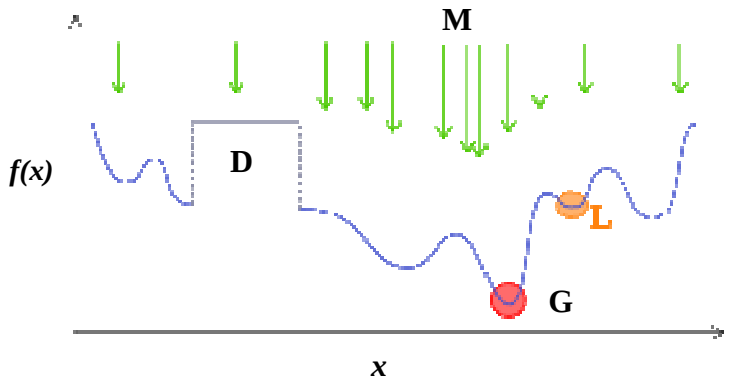


Fig. 1.4 : Echantillonnage probabiliste des méta-heuristiques (M). Elles tentent de trouver l'optimum global (G) d'un problème d'optimisation difficile (avec des discontinuités — D —, par exemple), sans être piégé par les optima locaux (L).

1.4.3 Propriétés des méta-heuristiques :

- Elles s'appliquent à des espaces de recherche discrète ou continue, elles sont souvent moins puissantes que des méthodes exactes sur certains types de problèmes.
- Ne nécessite pas le calcul de dérivées de fitness.
- Ce sont des méthodes stochastiques, qui explorent l'espace selon une composante aléatoire.

[1] Méta, du grec « au-delà » (comprendre ici «à un plus haut niveau»), heuristique, du grec heuriskein, qui signifie « trouver ».

- Ces méthodes sont inspirées de processus naturels.
- La colonie de fourmis est inspirée du comportement des animaux (éthologie) et plus précisément le comportement des insectes (entomologie).
- Les méta-heuristiques utilisent des paramètres pour guider et optimiser la façon dont l'espace est exploré. La valeur idéale de ces paramètres est en général inconnue. Cela peut d'ailleurs dépendre du problème particulier.
- Ces méthodes sont gourmandes en temps de calcul (mais on n'a pas d'autres choix). En revanche, elles se parallélisent assez facilement. [Chopard, 2006]
- Elles ne garantissent pas non plus la découverte de l'optimum global en un temps fini. Cependant, un grand nombre de problèmes réels n'est pas optimisable efficacement par des approches purement mathématiques, les méta-heuristiques peuvent alors être utilisées avec profit.

En dernière analyse, il est parfois possible que le choix de la représentation des solutions, ou plus généralement des méthodes associées à la méta-heuristiques, ait plus d'influence sur les performances que le type d'algorithme lui-même. En pratique, cependant, les méta-heuristiques se montrent plus puissantes que les méthodes de parcours exhaustif ou de recherche purement aléatoire [Wiki, 2012a].

1.5 Conclusion

On a voulu à travers ce chapitre de présenter quelques notions importantes, qui ont un lien avec les algorithmes de colonies de fourmis et de bien éclairer leur utilisation suivant ces notions et ces principes.

Chapitre 2

De la fourmi réelle à la fourmi artificielle

Chapitre 2 : De la fourmi réelle à la fourmi artificielle

2.1 Introduction

Les fourmis sont devenues dès lors une nouvelle source d'inspiration pour la conception de méthodes de résolution de problèmes complexes. De plus cette source d'inspiration n'est pas unique étant donné que les fourmis sont dotées d'une grande diversité de caractéristiques disjointes et de comportements collectifs variés. Une nouvelle classe d'algorithmes est alors apparue sous le nom « algorithmes de fourmis artificielles ». Leur popularité est due d'une part à la facilité de mise en œuvre et d'autre part à la complexité des fonctions réalisables.

Ce présent chapitre est consacré à présenter une introduction au monde des fourmis biologiques, ainsi les algorithmes reproduisant les facultés de l'insecte « fourmis » qui forment ainsi une classe de métaheuristique récemment proposée pour les problèmes de classification.

Ce terme "fourmi" est la pièrre angulaire de cette thèse. Derrière ce mot se profile plusieurs domaines: celui de la biologie ou plus précisément de la myrmécologie qui est l'étude du comportement naturel des fourmis, celui de la robotique qui utilise leur comportement pour concevoir des nouvelles machines, celui de l'informatique où ces créatures sont modélisées pour l'optimisation des problèmes réels où le tri et la classification des objets.

Ce présent chapitre est consacré à présenter une introduction au monde des fourmis naturelles ainsi les fourmis artificielles. A la fin une petite comparaison entre les fourmis réelles et les fourmis artificielles est faite pour sentir les résultats des recherches qui ont béneficier de la nature des insectes.

2.2 Les fourmis naturelles: [Monm, 2000a][Ouadfel, 2006]

2.2.1 Présentation de Fourmi:

Les fourmis (famille des Formicidae) sont des insectes sociaux formant des colonies (Fig. 2.2), appelées fourmilières (Fig.2.3), parfois extrêmement complexes, contenant de quelques dizaines à plusieurs millions d'individus. Les fourmis sont classées dans l'ordre des hyménoptères, c'est à dire des insectes dont les deux paires d'ailes sont membraneuses et fines [Ouadfel, 2006]. Les fourmis sont réparties en onze (11) sous-familles, approximativement 10 000 espèces.

2.2.2) Morphologie:

Les fourmis sont des insectes mesurant en moyenne de 0.01 à 3 centimètres, et pesant de 1 à 150 milligrammes. Elles ont un corps principalement de muscles enveloppés dans une carapace chitineuse très résistante. On peut observer que le corps de la fourmi est divisé en trois parties majeures bien reconnaissables :

- La tête qui est le support des antennes (récepteurs sensoriels extrêmement développés) et des mandibules (membres situés au niveau de la bouche qui se présentent sous forme de pinces dentées et puissantes).
- Le thorax qui permet la communication entre la tête et l'abdomen, soutenue par trois paires de pattes très longues et très fines qui permettent aux fourmis (tout comme aux araignées qui ne sont pas des insectes) de déplacer dans toutes les directions et dans toutes les positions possibles.
- L'abdomen, quand à lui, contient tout le système digestif et le moteur du système sanguin.

Même si la plupart des fourmis sont asexuées, certaines présentent un système reproductif mâle ou femelle. [Oudfel, 2006]

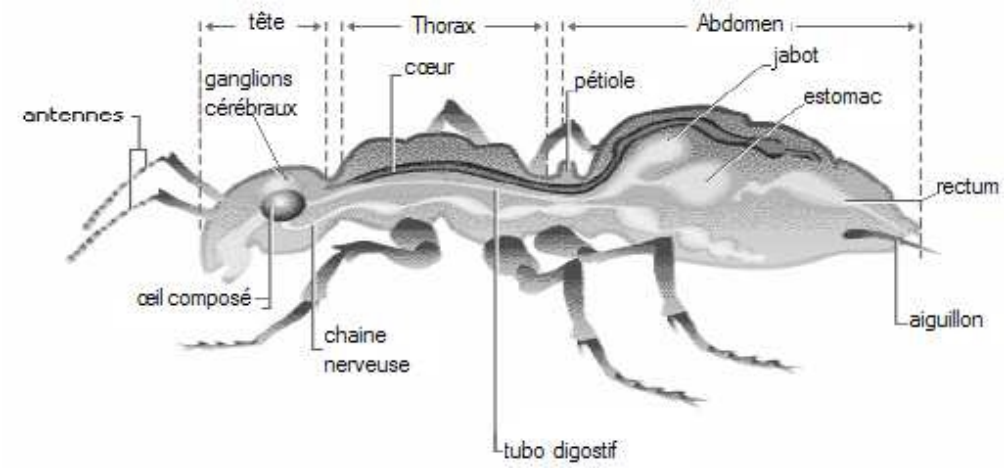


Fig. 2.1: Anatomie de la fourmi. [Ouadfel, 2006]

Les fourmis présentent un système nerveux peu développé comparé à celui de l'Homme (1 million de fois moins de neurones que le nôtre), un système sensoriel, digestif, circulatoire, ainsi que différentes glandes ayant des fonctions bien particulières. Ainsi, même si les fourmis apparaissent sous forme d'insectes minuscules, elles ont une organisation physiologique très développée et c'est celle-ci qui leur permet d'évoluer aisément dans leur environnement. De plus, elles présentent des lieux d'émission et de réception de messages chimiques, ce que nous allons développer prochainement [Ouadfel, 2006].

Fig. 2.2 Une colonie de fourmis

2.2.3 Fourmilière

La fourmilière classique (Fig.2.3) est constituée par un ensemble de cellules réunies entre elles par un réseau complexe de galeries qui peut être très important. Dans ces cellules, les fourmis déposent les œufs, les larves et les nymphes.

Les composants d'une fourmilière, numérotés sur la figure (Fig.2.3), sont [Harun, 2003][Focus, 1996] :

1. **Système de défense aérien (défense ant-aérienne)**: Lorsque les oiseaux, ennemis les plus redoutés des fourmis, s'approchent du nid, certains combattants tournent leurs ventres vers l'extérieur du nid et projettent de l'acide en direction des oiseaux.

2. **Serre (solarium incubateur)**: Les oeufs de la reine "mûrissent" dans cette pièce orientée vers le sud. Il y règne une température constante de 38°C.

3. **Entrée principale et entrées latérales**: Ces entrées sont gardées par les fourmis portières. En période de danger, elles bloquent les portes à l'aide de leurs têtes plates. Lorsque d'autres occupants du nid souhaitent entrer, ils frappent sur la tête des portiers avec leurs antennes selon un rythme déterminé et ces derniers les laissent passer.

S'ils venaient à oublier ce rythme, les portiers les tueraient sur le champ.

4. **Chambres prêtes (souche fondation)** : Si les fourmis trouvent un ancien nid dans lequel elles envisagent de s'installer, elles utilisent les chambres qui ont gardé une forme fonctionnelle. Elles gagnent ainsi un temps non négligeable en les incluant dans leur nouvelle structure.

5. **Cimetières de stockage (dépotoir cimetière)** : Les fourmis entreposent les écorces de grains non consommées et les corps des fourmis mortes dans ces pièces.

6. **Chambre des gardiens (salle de garde)** : Les fourmis-soldats qui logent ici sont en état d'alerte permanente. À la moindre sensation de danger, elles passent à l'action.

7. **Isolation extérieure (revêtement isolant)** : Leur système d'isolation, fait de branchages et de brindilles, constitue une protection efficace contre la chaleur, le froid et la pluie. Les ouvrières vérifient constamment l'état de l'isolation.

8. **Chambre d'allaitement (étable à puceron)** : Les fourmis nourrices produisent un doux liquide dans leurs abdomens qui sera utilisé par les fourmis en croissance.

9. **Dépôt de viande (grenier en viande)** : Les insectes, mouches, criquets et autres ennemis des fourmis sont stockés dans ce dépôt après avoir été tués.

10. **Dépôt de grain (grenier en grain)** : Les fourmis meunières transforment de gros morceaux de grains en petites particules qui leur serviront de nourriture en hiver.

11. **Garderie pour larves (crèche pour larves et nymphe)** : Les fourmis nourrices utilisent leur salive qui possède des propriétés antibiotiques pour protéger les bébés fourmis contre la maladie.

12. **Pièce hibernale (salle d'hébernation)** : Certaines fourmis hibernent de novembre à mai. Dès qu'elles sortent de l'hibernation, elles s'affairent à nettoyer cette pièce avant toute autre chose.

13. **Département de chauffage central (compost)** : Un mélange de morceaux de feuilles et de brindilles dégage une certaine chaleur en se décomposant. Ce phénomène augmente la température du nid d'environ 20 à 30 degrés.

14. **Pièce de couvée (couveuse pour les œufs)** : Les oeufs de la reine mère sont stockés dans cette chambre au fur et à mesure qu'ils sont pondus. À un temps voulu, ils sont retirés de cette chambre pour rejoindre la serre.

15. **Pièce royale (chambre royale)** : C'est ici que la reine pond ses oeufs. Ses assistants qui la nourrissent en permanence et nettoient la chambre y séjournent avec elle.

Ainsi, la fourmilière et ses environs constituent le centre de la vie communautaire. A l'intérieur de la fourmilière les tâches sont divisées entre les fourmis selon la spécialité de chacune d'elles. Les activités des communautés de fourmis sont caractérisées par un certain degré de division du travail souligné par une différenciation fonctionnelle et anatomique des individus. Une fourmilière peut abriter de 50.000 à plus de 1.000.000 individus bien différenciés tant au niveau physique qu'au niveau des comportements et des tâches à accomplir.

On les sélectionne en castes :

Les reines : Dans une fourmilière on trouve une ou plusieurs reines. Les reines sont nettement plus grosses que les autres fourmis et peuvent vivre jusqu'à dix ou quinze ans. Leur rôle se résume essentiellement à pondre des œufs et sont donc les fondatrices de nouvelles colonies.

Les ouvrières : Elles forment la majorité des habitants de la cité et se chargent de la défense et de l'entretien de la colonie, qui comprend la construction des galeries, les soins apportés aux jeunes, la quête de la nourriture, etc...

Les soldats : Ils sont plus massifs que les ouvrières, et possèdent souvent de grosses mandibules. Leur rôle est de défendre la fourmilière, et de transporter des charges lourdes. Mais certains, comme chez les fourmis "Grand Galop" Camponotus maculatus, participent aux soins des larves, et donnent à manger aux fourmis qui le demandent. Chez cette espèce on peut distinguer des formes intermédiaires entre la petite ouvrière grêle et le puissant soldat.

Le couvain : Il est constitué par les œufs, les larves et les nymphes. Au bout de quelques jours les œufs donnent naissance à des larves qui, bien nourries par les ouvrières pendant 15 jours à 3 semaines, se transforment en nymphes. Pendant la nymphose, la larve ne se nourrit plus. Son corps tout entier subit de profondes mutations internes et externes, qui vont faire d'elle, petit à petit, une fourmi.

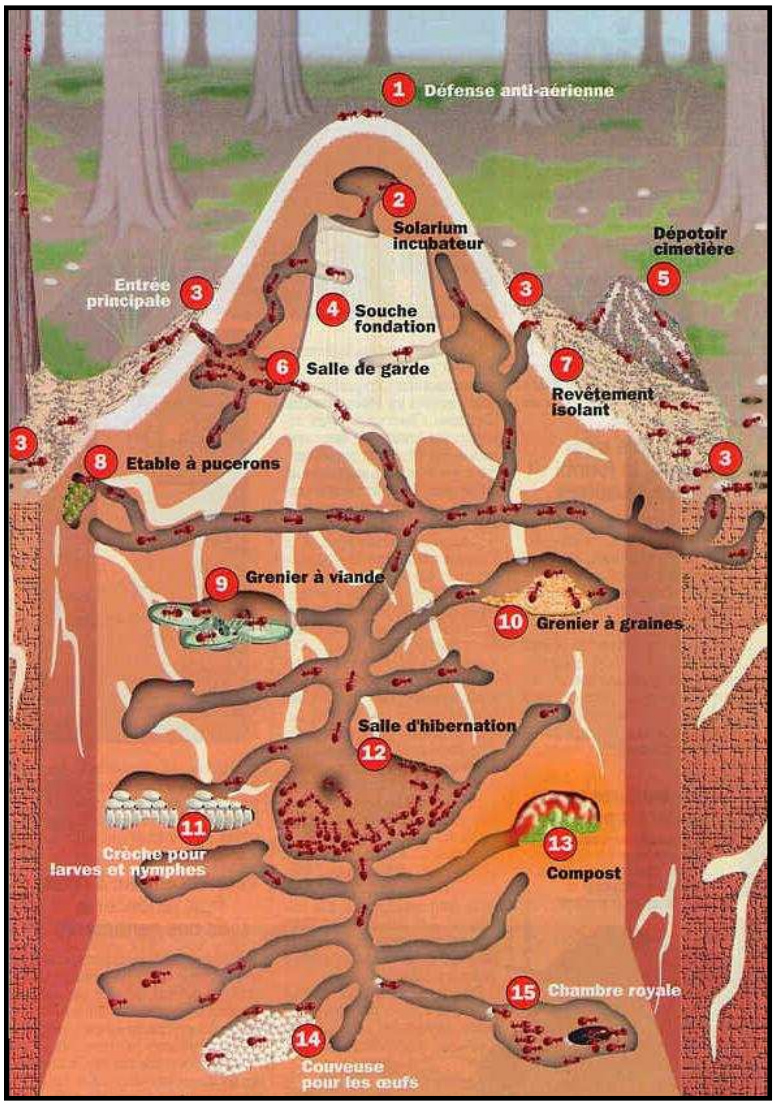


Fig. 2.3 Fourmilière

2.2.4 Les communications entre les fourmis

Les fourmis ont développé des mécanismes de communication très élaborés. Il a été défini douze types de réponse mettant en œuvre une forme de communication tels que : l'alarme, l'attraction simple, le recrutement,… [Monm, 2000]

La communication chez les fourmis peut-être de différentes natures : chimique, sonore, tactile, visuelle. [Ouadfel, 2006] Les principaux moyens de communication sont:

- **La communication tactile :** lorsque deux fourmis se rencontrent, elles procèdent à quelques attouchements d'antennes ou de pattes, et se reconnaissent aussitôt comme membres de la même fourmilière
- **La communication sonore :** les fourmis peuvent également utiliser des stimuli vibratoires comme moyen de communication. Elles frappent les parois de la

fourmilière avec leur abdomen pour prévenir les autres. Elles tapotent aussi leurs antennes pour se parler. Ce signal est utilisé en fonction de l'espèce comme signal de détresse en cas de danger, comme signal de qualité de l'alimentation pour le recrutement pour une source de nourriture, comme signal de demande d'aide en cas où la nourriture trouvée est de grande taille.

- **La communication visuelle :** développée chez certaines espèces, permet aux fourmis de retrouver leur chemin, guidées par des marqueurs visuels.

- **La communication chimique :** les fourmis sont équipées de glandes produisant des phéromones, substances chimiques volatiles et odorantes qu'elles peuvent sentir par leurs antennes. Ce signal chimique porte l'information à la fois sur l'espèce, la société mais aussi la caste et le stade de développement auxquelles appartiennent les fourmis rencontrées. C'est en sécrétant cette substance qu'une fourmi éclaireuse marque le chemin qu'elle a utilisé et revient avertir ses pairs de la présence de nourriture ou d'un danger, ainsi que de sa localisation. C'est le moyen de communication le plus efficace et plus utilisé par toutes les espèces de fourmis

2.2.4.1 Phéromone

Ce terme vient du verbe grec "pher" (porter) et de "hormone", comme l'étymologie l'indique, il signifie donc "porteur d'hormones". Les phéromones sont des substances produites par des glandes spécifiques. [Dréo, 2003][Ouadfel, 2006][Rida, 2006]

2.2.5 Mode de vie des fourmis [Deneubourg, 1991]

Les fourmis sont des insectes qui vivent en société. Elles peuplent tous les continents du monde et ont des comportements très variés suivant l'habitat dans lequel elles évoluent. Par exemple, les fourmis Cataglyphis qui ont pour origine probable le Sahara, ont de longues pattes et avancent rapidement à la surface des sols dénudés, ce qui leur permet de moins ressentir la chaleur. De plus elles ont la capacité de relever leur abdomen à la verticale pour que les rayons du soleil les brûlent moins.

Fig. 2.4 Fourmi Cataglyphis près de Maharès, Tunisie

En général les problèmes que les fourmis doivent surmonter quotidiennement sont les suivants:

- La recherche de nourriture. Cela demande une notion d'orientation pour la fourmi, qui doit pouvoir se souvenir de son site de chasse (notamment si la proie est trop grosse pour elle) et du lieu de son nid.

- L'exploitation d'une source de nourriture. Dans le cas où cette source contient un important volume de nourriture, elle doit pouvoir recruter les autres fourmis du nid, pour l'aider au transport de la nourriture.
- La reproduction et l'élevage de fourmi. Les fourmis trient les larves de leur couvain suivant la taille des larves.
- La construction d'un nid, la défense de ce dernier. Cela demande des moyens de communication, de partage des tâches et d'identification au sein de la colonie.

2.2.6 Principaux comportements de fourmis utilisés en intelligence artificielle

2.2.6.1Le comportement de tri d'objets

Beaucoup d'espèces de fourmis mettent naturellement en place des tris d'objets pour gérer leurs colonies. Ainsi, les Pheidole pallidula regroupent leurs morts, et les Leptothorax unifasciatus trient leurs larves en fonction de leur taille, ce qui leur permet d'optimiser la distribution de nourriture.

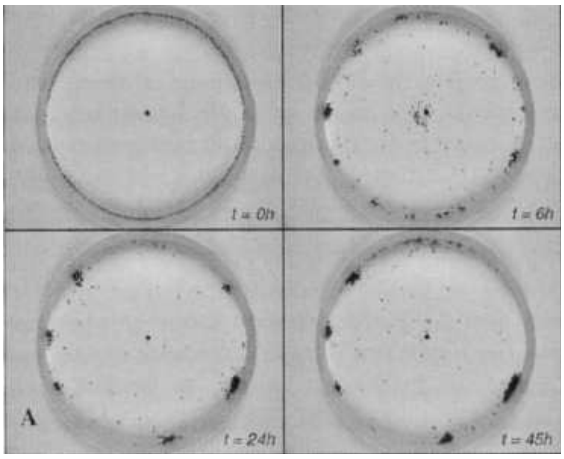


Fig. 2.5 Arène de 50 cm avec 400 cadavres. Expérience tirée de [Deneubourg, 1991]

2.2.6.2 La reconnaissance chimique

De nombreux algorithmes sont inspirés du comportement de masse des fourmis pour la recherche de nourriture : les phéromones de piste. Chaque fourmi dépose des phéromones à l'aide de son abdomen pour tracer des pistes et suivre un chemin. Il en résulte un phénomène auto-catalytique, en effet, plus une piste est suivie et plus il y a de phéromones et plus il y a de fourmis. Notons que ces phéromones s'évaporent au cours du temps. Les travaux de Deneubourg et al [Deneubourg, 1991] montrent que le plus court chemin pour atteindre une source de nourriture ne finira pas être emprunté par toutes les fourmis puisque les phéromones sont déposés plus vite sur ce plus court chemin. La reconnaissance chimique par les phéromones est utilisée dans de nombreuses situations dans lesquelles les fourmis peuvent être confrontées. Par exemple, lorsqu'une fourmi se sent en danger ou a trouvé une proie vivante trop grosse pour elle, elle rejette un nombre important de phéromones, pour que ses congénaires puissent remarquer le signal de loin. Les autres fourmis s'approchent donc de la proie, mais le taux de phéromones étant trop fort pour elles, elles restent en retrait. Un nombre important de fourmis va donc

s'agglutiner à proximité et lorsque que le taux de phéromones devient plus faible (dû à l'évaporation), elles attaquent tout ensemble la proie.
Enfin, chez les insectes sociaux, la défense implique la fermeture coloniale, autrement dit l'hermétisme d'une colonie à tout individu étranger. Cette fermeture remplit de fait une fonction analogue à celle de notre système immunitaire. Dans le cas des fourmis, leur odeur est identique à celui de leur colonie et du nid à un instant donné. La mise en place de l'odeur coloniale fait intervenir une glande particulière qui occupe environ les deux tiers du volume total de la tête de chaque fourmi.

2.2.6.3 L'auto-assemblage chez les fourmis

La notion d'auto-assemblage se remarque chez certaines fourmis, notamment les fourmis tisserandes, qui construisent leurs nids dans les arbres. En effet, elles mettent les feuilles bords à bords en les cousant et sans les couper car ces feuilles doivent rester vivantes pour la solidité du nid. Pour rapprocher ces feuilles, les fourmis tirent chaque bord avec leurs mandibules. Si les feuilles sont trop éloignées, elles forment une chaîne, c'est-à-dire qu'elles s'accrochent entre elles (elles s'auto-assemblent) puis courbent et rapprochent les feuilles. La couture est ensuite faite avec la soie collante produite par les larves.

Fig. 2.6 Fourmis tisserandes *Oecophylla*, Australie

Les fourmis ont inspiré les scientifiques dans plusieurs domaines, l'optimisation et la classification. Cette étude de l'existant est plus particulièrement portée sur les modèles de classifications, bien qu'un survol des autres domaines soit effectué.

2.3 Les fourmis artificielles

L'étude sur le comportement des fourmis a donné une naissance à plusieurs nouvelles méthodes de résolution de problèmes.

Certain mécanismes de résolution collective de problèmes chez les fourmis ont été transformés par des informaticiens en méthodes utiles pour l'optimisation et le contrôle qui peuvent s'appliquer aujourd'hui à tout un ensemble de problèmes scientifiques et techniques [Bonabeau, 2000][Monm, 2000].

Les fourmis artificielles sont des agents artificiels et par facilité son nom de fourmi. La programmation informatique moderne permet de faire fonctionner en parallèle divers « agents », par exemple des agents fourmis qui interagissent entre eux et avec leur environnement. Chaque agent répond en fonction d'une ou d'un petit nombre de règles simples, sans qu'il soit nécessaire de faire appel à des capacités cognitives élaborés. Les

fourmis modélisées sont alors des agents réactifs et on parle alors d'intelligence en essaim [Bonabeau, 1999] [Bonabeau, 2000].

La fourmi artificielle peut prendre la forme d'un processus informatique, qui possède donc ses propres zones de mémoire et ses instructions de fonctionnement. L'intérêt de la modélisation et la simulation informatique pour le biologiste est évident : cela permet de tester des modèles facilement (par rapport à l'effort expérimental de manipuler de vraies fourmis), de chercher une explication à des phénomènes émergents ou de tester la capacité de prédiction d'un modèle.

Quand il s'agit de résoudre des problématiques « humaines », la fourmi artificielle, va perdre encore de ce qui la rattachait à son modèle naturel. Si cela suffit de considérer comme un point dans un espace de recherche auquel on s'intéresse, on s'abstiendra bien de lui fournir des pattes, un thorax, des antennes, etc. Le lien entre le point et la vraie fourmi ne sera alors peut être plus qu'une lointaine similitude dans la stratégie de déplacement du point et le modèle de comportement de la fourmi dans son environnement naturel [Lenoir, 2009].

Enfin, après un constat de la suprématie des fourmis dans les environnements naturels nous pourrions tenter la même analyse dans le domaine des fourmis artificielles. C'est ainsi que, si l'on observe le nombre d'articles publiés dans des journaux ou conférences scientifiques dans les domaines de l'informatique, la robotique ou des mathématiques appliquées, on constate, même avec un relevé incomplet, une augmentation significative du nombre abordant la notion de fourmi artificielle (fig. 2.7).

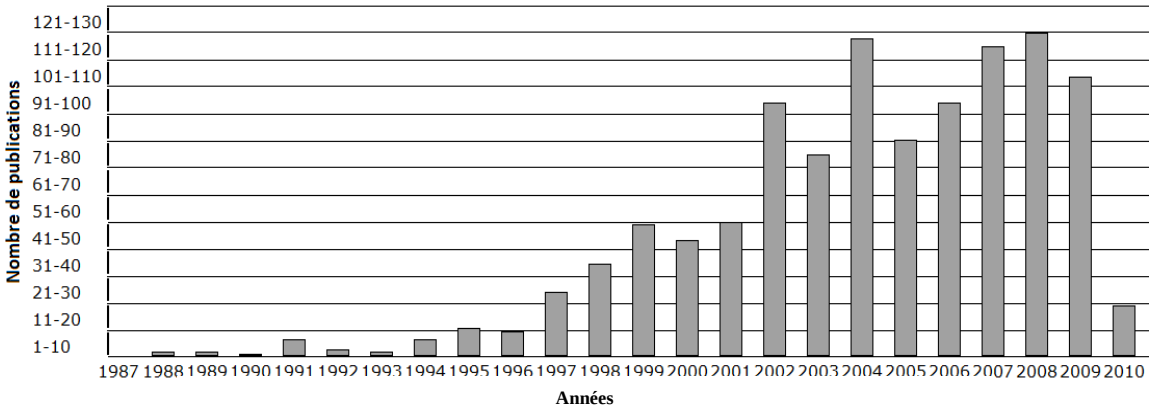


Fig. 2.7. Evolution du nombre de publications repérées qui abordent le thème des fourmis artificielles

2.3.1 Les algorithmes de colonies de fourmis :

Les algorithmes de colonies de fourmis sont considérés comme des **métaheuristiques à population**, où chaque solution est représentée par une fourmi se déplaçant sur l'espace de recherche. Les fourmis marquent les meilleures solutions par des phéromones, et tiennent compte des marquages précédents pour optimiser leurs recherches. [Bonabeau, 1999] [Dreo, 2003] [Dorigo, 2006]

On peut les considérer comme des algorithmes multi-agents probabilistes, utilisant une distribution de probabilité implicite pour effectuer la transition entre

chaque itération. Dans leurs versions adaptées à des problèmes combinatoires, ils utilisent une construction itérative des solutions.

D'après certains auteurs, ce qui différencierait les algorithmes de colonies de fourmis d'autres métaheuristiques proches (telles que les algorithmes à estimation de distribution ou l'optimisation par essaim particulaire) serait justement son aspect constructif. En effet, dans les problèmes combinatoires, il est possible que la meilleure solution finisse par être trouvée, alors même qu'aucune fourmi ne l'aura éprouvée effectivement.

Le comportement collectif des insectes sociaux reste une source d'inspiration pour les chercheurs. La grande diversité d'algorithmes (pour l'optimisation ou non) se réclamant de l'auto-organisation dans les systèmes biologiques a donné lieu au concept d'« intelligence en essaim », qui est un cadre très général, dans lequel s'inscrivent les algorithmes de colonies de fourmis.

L'algorithme de colonies de fourmis a été à l'origine surtout utilisé pour produire des solutions quasi-optimales au problème du voyageur de commerce, puis, plus généralement, aux problèmes d'optimisation combinatoire. On observe que depuis ses débuts son emploi s'est étendu à plusieurs domaines, depuis l'optimisation continue jusqu'à la classification ou encore le traitement d'image.

2.3.1.1 Historique des algorithmes de colonies de fourmis :

- 1959, Pierre-Paul Grassé invente la théorie de la stigmergie pour expliquer le comportement de construction du nid chez des termites ;
- 1983, Deneubourg et ses collègues étudient le comportement collectif des fourmis ;
- 1988, Moyson et Manderick présentent un article sur l'auto-organisation chez les fourmis ;
- 1989, travaux de Goss, Aron, Deneubourg et Pasteels, sur le **comportement collectifs des fourmis Argentines**, qui donneront l'idée des ***algorithmes de colonies de fourmis*** ;
- 1989, implémentation d'un modèle de comportement de recherche de nourriture par Ebling et ses collègues ;
- 1991, M. Dorigo propose l'Ant System dans sa thèse de doctorat (qui ne sera publiée qu'en 1992). Il fait paraître, avec V. Maniezzo et A. Colorni, un rapport technique, qui sera publié cinq ans plus tard ;
- 1991, Une des premières œuvres réalisées par Deneubourg en 1991 était liée aux agents ressemblant à des fourmis qui sont autorisés à déplacer des objets au hasard sur une grille à deux dimensions grille pour objectif de partitionnement ;
- 1994, Lumer et Faieta propose un algorithme de classification non supervisée appliquée à l'analyse des données et en 1995, ils ont appliqué leur algorithme pour l'analyse exploratoire de base de données comme une alternative à la caractéristique des classes ;
- 1995, Bilchev et Parmee publient la première tentative d'**adaptation aux problèmes continus** ;
- 1996, publication de l'article sur l'Ant *System* ;
- 1996, Stützle et Hoos inventent le ***MAX-MIN Ant Sytem*** ;
- 1997, Dorigo et Gambardella publient le ***Ant Colony System*** ;

- 1997, Schoonderwoerd et ses collègues concoivent la première application aux réseaux de télécommunications ;
- 1997, Martinoli et ses collègues s'inspirent des algorithmes de colonies de fourmis pour le contrôle de *robots* ;
- 1998, Dorigo lance la première conférence dédiée aux algorithmes de colonies de fourmis ;
- 1998, Stützle propose les premières ***implémentations parallèles*** ;
- 1999, Bonabeau et ses collègues font paraître un livre traitant principalement des *fourmis artificielles* ;
- 1999, premières applications pour le routage de véhicule, l'assignement quadratique, le sac à dos multi-dimensionnel ;
- 1999, L'algorithme Antclass est une partie de la thèse de Nicolas Monmarché en 2000. Il reprend les idées des algorithmes de classification à base de fourmis et ajoute des contraintes pour améliorer les performances.
- 2000, numéro spécial d'une revue scientifique sur les algorithmes de colonies de fourmis ;
- 2000, premières applications à l'ordonnancement, l'ordonnancement séquentiel, la satisfaction de contraintes ;
- 2000, Gutjahr donne la première preuve de ***convergence*** pour un algorithme de colonies de fourmis ;
- 2001, première utilisation des algorithmes de colonies de fourmis par des entreprises (Eurobios et AntOptima) ;
- 2001, Iredi et ses collègues publient le premier algorithme ***multi-objectif*** ;
- 2002, Tsai, Wu et Tasi Crée (ACODF) par la modification de (ACO) par l'ajout du concept ***recuit simulé*** (***simulated annealing***) ;
- 2002, premières applications à la conception d'emploi du temps, les réseaux bayésiens ;
- 2002, Bianchi et ses collègues proposent le premier algorithme pour problème stochastique ;
- 2002, L'algorithme ACLUSTER de Ramos et Merelo, est une évolution des modèles de Lumer et Faieta [Lumer, 1994] et de Deneubourg [Deneubourg, 1991]. Il comprend deux principaux changements inspirés de la biologie : l'ajout des phéromones qui permet à l'algorithme de base de Lumer et Faieta d'accélerer la convergence (de la même manière que l'ajout de mémoire aux fourmis), et la mise en place d'un seuillage pour la prise et le dépôt d'objets par rapport à l'intensité d'objets dans le voisinage. ACLUSTER est employé pour classifier des images et du texte.
- 2004, Pun, Chu, Roddik et Su, (CACO) pour la classification des données ;
- 2004, Zlochin et Dorigo montrent que certains algorithmes sont équivalents à la descente stochastique de gradient, l'entropie croisée et les algorithmes à estimation de distribution ;
- 2005, premières applications au repliement de protéines. [Scho, 2004]
- 2006, Ouadfel Salima. Contributions à la Segmentation d'images basées sur la résolution collective par colonies de fourmis artificielles [Ouadfel, 2006].

2.3.2 Similarités et différences avec les fourmis réelles :

Les fourmis virtuelles (artificielles) ont une double nature. D'une part, elles modélisent les comportements abstraits de fourmis réelles, et d'autre part, elles peuvent

être enrichies par des capacités que ne possèdent pas les fourmis réelles, afin de les rendre plus efficaces que ces dernières. Nous allons maintenant synthétiser ces ressemblances et différences. [Costanzo, 2006]

2.3.2.1 Points communs :

- ***Colonie d'individus coopérants :*** Comme pour les fourmis réelles, une colonie virtuelle est un ensemble d'entités non-synchronisés, qui se rassemblent ensemble pour trouver une "bonne" solution au problème considéré. Chaque groupe d'individus doit pouvoir trouver une solution même si elle est mauvaise.
- **Pistes de phéromones *:*** Ces entités communiquent par le mécanisme des pistes de phéromone. Cette forme de communication joue un grand rôle dans le comportement des fourmis : son rôle principal est de changer la manière dont l'environnement est perçu par les fourmis, en fonction de l'historique laissé par ces phéromones.
- **Évaporation des phéromones *:*** La méta-heuristique ACO comprend aussi la possibilité d'évaporation des phéromones. Ce mécanisme permet d'oublier lentement ce qui s'est passé avant. C'est ainsi qu'elle peut diriger sa recherche vers de nouvelles directions, sans être trop contrainte par ses anciennes décisions.
- **Recherche du plus petit chemin *:*** Les fourmis réelles et virtuelles partagent un but commun : recherche du plus court chemin reliant un point de départ (le nid) à des sites de destination (la nourriture).
- **Déplacement locaux :** Les vraies fourmis ne sautent pas des cases, tout comme les fourmis virtuelles. Elles se contentent de se déplacer entre sites adjacents du terrain.
- **Choix aléatoire lors des transitions *:*** Lorsqu'elles sont sur un site, les fourmis réelles et virtuelles doivent décider sur quel site adjacent se déplacer. Cette prise de décision se fait au hasard et dépend de l'information locale déposée sur le site courant. Elle doit tenir compte des pistes de phéromones, mais aussi du contexte de départ (ce qui revient à prendre en considération les données du problème d'optimisation combinatoire pour une fourmi virtuelle).

2.3.2.2 Différences :

Les fourmis virtuelles possèdent certaines caractéristiques que ne possèdent pas les fourmis réelles :

- **Elles vivent dans un monde non-continu *:*** Leurs déplacements consistent en des transitions d'état.
- **Mémoire (état interne) de la fourmi :** Les fourmis réelles ont une mémoire très limitée. Tandis que nos fourmis virtuelles mémorisent l'historique de leurs actions. Elles peuvent aussi retenir des données supplémentaires sur leurs performances.
- **Nature des phéromones déposées :** Les fourmis réelles déposent une information physique sur la piste qu'elles parcourent, là où les fourmis virtuelles modifier des informations dans les variables d'états associées au problème. Ainsi, l'évaporation des phéromones est une simple décrémentation de la valeur des variables d'états à chaque itération.
- **Qualité de la solution *:*** Les fourmis virtuelles déposent une quantité de phéromone proportionnelle à la qualité de la solution qu'elles ont découverte.
- **Retard dans le dépôt de phéromone *:*** Les fourmis virtuelles peuvent mettre à jour les pistes de phéromones de façon non immédiate : souvent elles attendent d'avoir

terminé la construction de leur solution. Ce choix dépend du problème considéré bien évidemment.

- **Capacités supplémentaires :** Les fourmis virtuelles peuvent être pourvues de capacités artificielles afin d'améliorer les performances du système. Ces possibilités sont liées au problème et peuvent être :

  1. ***l'anticipation :*** la fourmi étudie les états suivants pour faire son choix et non seulement l'état local.
  2. ***le retour en arrière :*** une fourmi peut revenir à un état déjà parcouru car la décision qu'elle avait prise à cet état a été mauvaise.

2.3.3 Avantages et inconvénients des fourmis artificielles:

- ***Avantage :***
  - Très grande adaptabilité.
  - Parfait pour les problèmes basés sur des graphes.
- ***Inconvénients :***
  - Un état bloquant peut arriver.
  - Temps d'exécution parfois long.
  - Ne s'applique pas à tous type de problèmes. [Monm, 2000a]

2.4 Application des algorithmes

2.4.1 Application aux problèmes d'optimisation

Les algorithmes d'optimisation sont regroupés sous le terme général **méta-heuristique** « optimisation par les colonies de fourmis (OCF) » ou « Ant Colony Optimization (ACO) » [Dorigo, 2004]. D'une manière générale, l'utilisation de OCF nécessite de choisir une représentation formelle du problème d'optimisation à traiter et de définir le processus de construction de solutions par les fourmis en utilisant cette représentation. [Ouadfel, 2006]

2.4.2 Application au tri d'objets:

Deneubourg apparaît comme le pionnier dans le domaine du tri automatique d'objets en utilisant des fourmis artificielles

2.4.3 Application à la classification

Le pas qui sépare le tri d'objets de la classification a ensuite été franchi dans l'article de Monmarché. Il existe plusieurs algorithmes de classification automatique basée sur les fourmis artificielles. On peut aussi introduire un modèle à base de fourmis pour la classification utilisant le système d'identification chimique des fourmis.

2.4.4 Application en traitement d'images

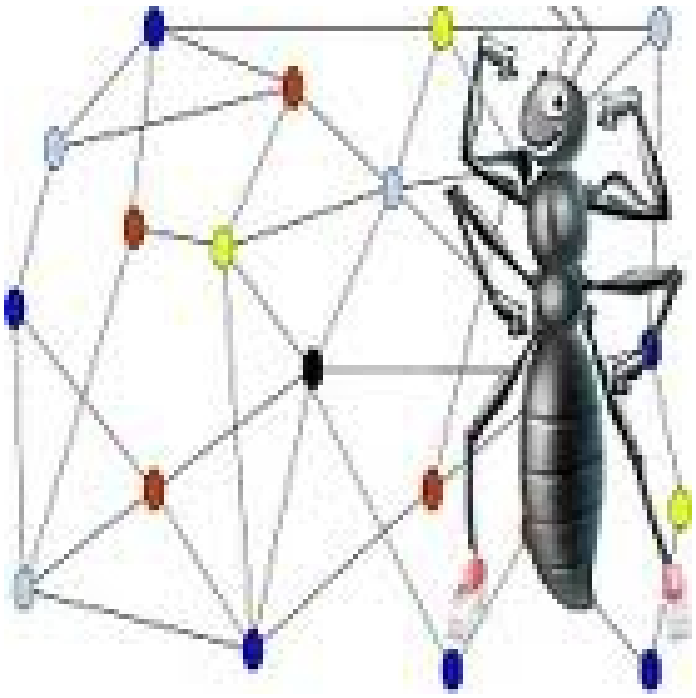
Des études plus récentes ont étendu les algorithmes de colonies de fourmis aux problèmes de la classification pour les appliquer au domaine du traitement d'images tel que les travaux d'Ouadfel dans son thése [Ouadfel, 2006].

2.5 Conclusion

Ce chapitre a présenté un aperçu sur les fourmis naturelles ainsi les fourmis artificielles. On a détaillé le passage entre le réelle et l'artificielle qui a abouti à la naissance des "algorithmes de colonies de fourmis". Le chapitre se termine avec un petit historique sur ces algorithmes ainsi une comparaison entre les fourmis réelles et les fourmis artificielles.

Chapitre 3

Le problème d'optimisation



Chapitre 3 : Le problème d'optimisation

3.1. Introduction

L'optimisation est un sujet central en informatique, plusieurs problèmes d'aide à la décision pouvant en effet être décrits sous la forme de problèmes d'optimisation.

L'objectif de l'optimisation est de déterminer une solution qui minimise (ou maximise) une fonction appelée dans la littérature ***fonction objective*** tout en vérifiant un certain nombre de contraintes et le but final est de trouver l'optimum global (le maximum ou le minimum global).

La méthode de programmation non linéaire a été la première méthode à connaître un essor remarquable, attirent ainsi l'attention des chercheurs et des ingénieurs ; les solutions qu'elles offrent couvrent un large champ d'application.

Les problèmes d'apprentissage, par exemple l'apprentissage des réseaux de neurones, la recherche du plus court chemin, ou encore le problème de production dans les systèmes électriques de puissance qui, à partir de plusieurs unités de production d'alimenter plusieurs charges, obligeant l'opérateur à décider comment répartir la charge entre les différentes unités nécessitent une phase d'optimisation.

Le problème de classification qui est abordé dans notre travail de thèse est aussi un problème d'optimisation : minimiser la distance intra-classe et maximiser la distance inter-classe.

Dans la vie, nous sommes fréquemment confrontés à des problèmes d'optimisation plus au moins complexes. Cela peut commencer au moment où l'on tente de ranger notre bureau, de placer nos mobiliers, de gérer notre espace dans la maison de minimiser nos trajets en voiture et aller jusque à un processus industrielle on définit alors une fonction objective, que l'on cherche à optimiser par rapport à tous les paramètres concernés (voir **Fig.3.3)**.

En pratique l'objectif n'est pas d'obtenir un optimum absolu, mais seulement une bonne solution, et la garantie de l'inexistence d'une solution sensiblement meilleure. Pour atteindre cet objectif au bout d'un temps de calcul raisonnable, il est nécessaire d'avoir recours à des méthodes appelées « heuristique ». Ces dernières produisant des solutions proches de l'optimum et la plupart d'entre elles sont conçues pour un type de problème donné. D'autre au contraire, appelés « méta heuristique », sont capables de s'adapter aux différents types de problèmes.

Lorsqu'un nouveau problème se pose en ingénierie, il faut parfois définir de nouvelles méthodes de résolution car les techniques existantes ne sont pas précisément adaptées au cas traité. Ainsi, lorsque l'on veut inventer une nouvelle méthode de résolution de problème, il faut souvent une source d'inspiration. Celle-ci peut être totalement imaginaire et n'est pas obligée de faire référence au monde réel comme par exemple des méthodes mathématiques abstraites ou peut au contraire être issue de la modélisation des systèmes complexes naturels. Il s'agit dans ce dernier cas de copier et d'adapter les concepts mis en œuvre par le monde du vivant pour la résolution de problèmes d'optimisation. Les recherches sur les comportements collectifs des insectes sociaux fournissent aux informaticiens des méthodes puissantes pour la conception d'algorithmes d'optimisation combinatoire difficile.

En recherche opérationnelle, et plus précisément dans le domaine de l'optimisation difficile, la majorité des méthodes sont inspirées par de telles études, et notamment par la biologie. Le fait que la biologie étudie souvent des systèmes présentant des comportements dits "intelligents" n'est pas étranger au fait qu'ils soient modélisés, puis transposés dans le cadre de problèmes "réels". On parle parfois d'intelligence artificielle biomimétique pour désigner de telles approches.

Dans le cadre de l'optimisation, cette approche a donné lieu à la création de nouvelles méta-heuristiques. Les méta-heuristiques forment une famille d'algorithmes d'optimisations visant à résoudre des problèmes d'optimisation difficile, pour lesquels on ne connaît pas de méthode classique plus efficace. Elles sont généralement utilisées comme des méthodes génériques pouvant optimiser une large gamme de problèmes différents, sans nécessiter de changements profonds dans l'algorithme employé. Les algorithmes de colonies de fourmis, les algorithmes génétiques et l'Asynchrone PSO forment ainsi une classe de méta-heuristique récemment proposée pour les problèmes d'optimisation difficile.

Dans les années 80 du vingtième siècle, le développement rapide de l'outil informatique a permis d'élaborer d'autres méthodes :

- Des méthodes de programme successives.
- Des méthodes de lagrangienne augmenté.
- Des méthodes de programmation quadratique mentionnent les méthodes de Newton et Quasi –Newton.

Les spécialistes de l'optimisation combinatoire ont ensuite orienté leur recherche vers le développement des méthodes stochastique tel que : le recuit simule, la rechercher tabou et les algorithmes évolutionnistes.

D'autres méta-heuristiques plus avancées ont vues le jour ces dernières années et ont nécessitées de les faire essayer à la résolution des problèmes existants pour permettre d'améliorer les résultats des anciennes méthodes. Comme exemple de ces méta-heuristiques :

Algorithme de colonies de fourmis : *ACO* (Ant Colony Optimization), Parallel Asynchrone PSO, algorithmes génétiques et d'autres.

Ce chapitre décrit tout d'abord le cadre de l'optimisation ainsi l'optimisation difficile et on commence par une présentation détaillée sur les méta-heuristiques de colonies de fourmis".

3.2 Définition du problème d'optimisation

Un problème d'optimisation au sens général est défini par un ensemble de solutions possibles ***S***, dont la qualité peut être décrite par une fonction objective ***f***. On cherche alors à trouver la solution S^* possédant la meilleure qualité $F(S^*)$ (par la suite, on peut chercher à minimiser ou à maximiser $F(S)$).

Un problème d'optimisation peut présenter des contraintes d'égalité (ou d'inégalité) sur S, être dynamique si ***f(S***) change avec le temps ou encore multi-objectif si plusieurs fonctions objectives doivent être optimisées [Allaoua , 2009].

Formulation d'un problème d'optimisation

Sans perte de généralité, nous nous plaçons désormais dans le cadre d'un problème de minimisation.

Un problème d'optimisation ***(P)*** de type « minimisation » et de dimension ***n*** peut être Formulé de façon générale comme suite :

$$(P)\begin{cases} \min f_{obj}(x) \\ x \in D \\ g_i(x) \le 0 \quad 1 \le i \le n \\ h_j(x) = 0 \quad 1 \le j \le n \end{cases} \qquad \text{(Eq.3.1)}$$

Où :
x : Est un vecteur à n composante représentant les variables objets du problème.
D : Est un espace des paramètres (ou espace de recherche).
f_{obj} : Critère à minimiser.
$g_i(x) \le 0$: Les contraintes d'inégalité, leur *domaine de validité. C*es contraintes permettent de restreindre le domaine de validité à *D*. Notons que ces contraintes peuvent ne pas exister, auquel cas nous parlons d'optimisation sans contrainte.
$h_j(x)=0$: les contraintes d'égalité
Un point x^* de l'espace est un minimum local si $\forall x \in$ D tel que:

$$d(x,x^*) \le \varepsilon, \qquad f_{obj}(x^*) \le f_{obj}(x) \qquad \text{(Eq.3.2)}$$

Où : $d(x,x^*$ désigne la distance entre les points ***x*** et ***x****.
La figure (**Fig.3.1**) présente, à titre d'exemple, une distribution possible des optimums d'une fonction objectif.

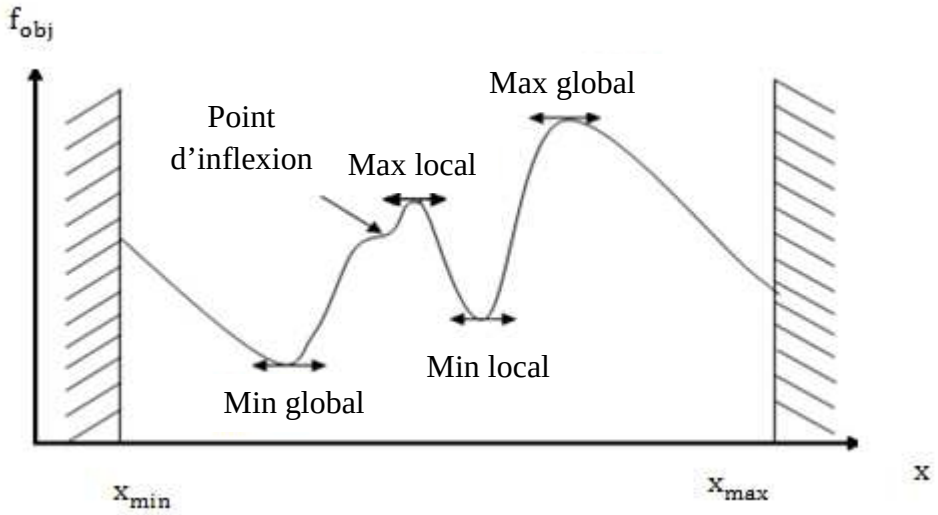


Fig. 3.1 Point singuliers d'une fonction

3.3 Les techniques d'optimisation [Dréo, 2003]

Les techniques d'optimisation mono-objective se divisent en deux grandes parties : l'optimisation discrète (combinatoire) et l'optimisation continue (**Fig.3.2)**.

Pour l'optimisation combinatoire, on a recours aux méthodes approchées, lorsqu'on est confronté à un problème difficile ; dans ce cas, le choix est parfois possible entre une heuristique, entièrement dédiée au problème considéré, et une méta-heuristique ;

Pour l'optimisation continue, on sépare le cas linéaire du cas non linéaire, où l'on retrouve le cadre de l'optimisation difficile.

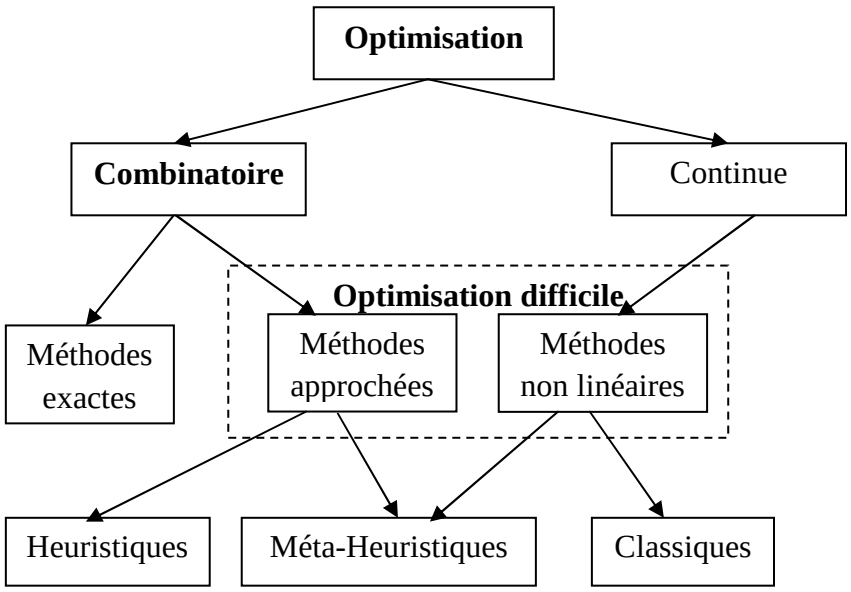


Fig.3.2 Classification générale des méthodes d'optimisation mono-objective

3.3.1 Un problème combinatoire :

Un problème combinatoire est toute situation dont on cherche d'avoir une solution tout en respectant la présence d'un ensemble de contraintes. La solution est de faire combiner ces contraintes ensemble d'une manière qu'on maximise quelques-uns et on minimise les autres, ces contraintes ont une caractéristique primordiale, c'est que chaque contrainte influe sur les autres soit quand on minimise sa valeur ou on la maximise, dans un autre terme on dit que les contraintes sont conflictuelles. Par exemple, la figure **(Fig.3.3)** présente une situation de problème combinatoire: où on veut acheter une voiture dans la mode et en même temps avec un prix raisonnable qui ne peut pas dépasser certaine limite. Si on maximise la première contrainte (une bonne voiture) on va avoir un prix maximale, dans le contraire on va aboutir à une mauvaise voiture mais avec un prix minimale dans les limites; on constate dans cet exemple que c'est difficile d'arranger ces deux contraintes dans nos besoins [Semet, 2003] [Dorigo, 2004] [Berro, 2001] [Helton, 2005].

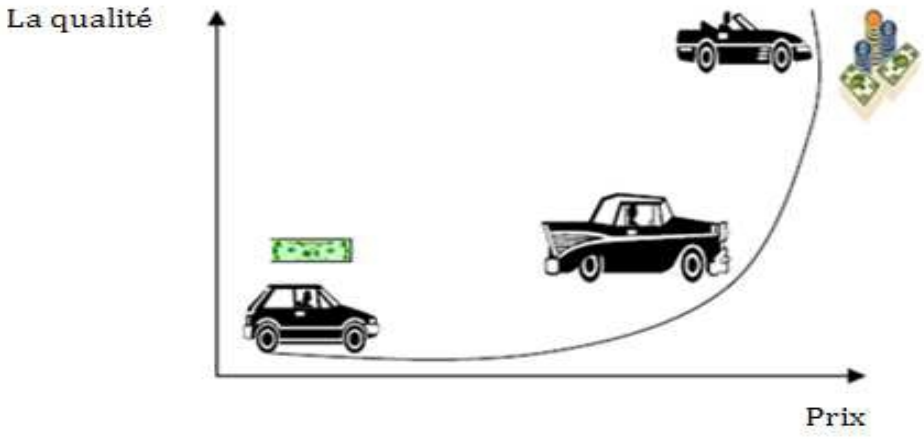


Fig.3.3 Une figure illustrant un problème combinatoire

3.3.1.1Optimisation difficile

Certains problèmes d'optimisation combinatoire demeurent cependant hors de portée des méthodes exactes. Un certain nombre de caractéristiques peuvent en effet être problématiques, comme l'absence de convexité stricte (multi-modalité), l'existence de discontinuités, une fonction non dérivable, présence de bruit,…etc.

Dans de tels cas, le problème d'optimisation est dit "difficile", car aucune méthode exacte n'est capable de le résoudre exactement en un temps "raisonnable", on devra alors faire appel à des heuristiques permettant une optimisation approchée.

3.4 Optimisation par les algorithmes de Colonies de fourmis

3.4.1 Introduction

Les fourmis sont capables de résoudre collectivement des problèmes complexes, comme trouver le plus court chemin entre deux points dans un environnement accidenté. Pour cela, elles communiquent entre elles de façon locale et indirecte, grâce à une hormone volatile, appelée phéromone : au cours de sa progression, une fourmi laisse derrière elle une trace de phéromone qui augmente la probabilité que d'autres fourmis passant à proximité choisissent le même chemin [Deneubourg, 1990] [Dorigo, 2004].

Ce mécanisme de résolution collective de problèmes est à l'origine des algorithmes à base de fourmis artificielles. Ces algorithmes ont été initialement proposés dans [Dorigo, 1992] [Dorigo, 1996], comme une approche multi-agents pour résoudre des problèmes d'optimisation combinatoire. L'idée est de représenter le problème à résoudre sous la forme de la recherche d'un meilleur chemin dans un graphe, appelé graphe de construction, puis d'utiliser des fourmis artificielles pour rechercher de bons chemins dans ce graphe. Le comportement des fourmis artificielles est inspiré des fourmis réelles : elles déposent des traces de phéromone sur les composants du graphe de construction et elles choisissent leurs chemins relativement aux traces de phéromone précédemment déposées ; ces traces sont évaporées au cours du temps.

Intuitivement, cette communication indirecte via l'environnement — connue sous le nom de stigmergie — fournit une information sur la qualité des chemins empruntés afin d'attirer les fourmis et d'intensifier la recherche dans les itérations futures vers les zones correspondantes de l'espace de recherche. Ce mécanisme d'intensification est combiné à un mécanisme de diversification, essentiellement basé sur la nature aléatoire des décisions prises par les fourmis, qui garantit une bonne exploration de l'espace de recherche. Le compromis entre intensification et diversification peut être obtenu en modifiant les valeurs des paramètres déterminant l'importance de la phéromone.

Les fourmis artificielles ont aussi d'autres caractéristiques, qui ne trouvent pas leur équivalent dans la nature. En particulier, elles peuvent avoir une mémoire, qui leur permet de garder une trace de leurs actions passées. Dans la plupart des cas, les fourmis ne déposent une trace de phéromone qu'après avoir effectué un chemin complet, et non de façon incrémentale au fur et à mesure de leur progression. Enfin, la probabilité pour une fourmi artificielle de choisir un arc ne dépend généralement pas uniquement des traces de phéromone, mais aussi d'heuristiques dépendantes du problème permettant d'évaluer localement la qualité du chemin.

Ces caractéristiques du comportement des fourmis artificielles définissent la «méta-heuristique d'optimisation par une colonie de fourmis» ou «Ant Colony Optimization (*ACO*) metaheuristic» [Dorigo, 1999a] [Dorigo, 1999b] [Dorigo, 2004]. Cette méta-heuristique a permis de résoudre différents problèmes d'optimisation combinatoire, comme par exemple le problème du voyageur de commerce [Dorigo, 1997b], le problème

d'affectation quadratique [Gambardella, 1999] ou le problème de routage de véhicules [Bullnheimer, 1999].

Dans la suite nous décrirons en détail chacun des modèles de fourmis artificielles ainsi que les différents algorithmes qui lui sont associés [Goss, 1989] [Dorigo, 1997a] [Stützle, 1999] [Taillard, 1999] [Dorigo, 1997b] [Stützle, 2000] [Costa, 1997] [Dudot, 2005] et on termine par la formalisation et propriétés d'un algorithme de colonies de fourmis pour l'optimisation.

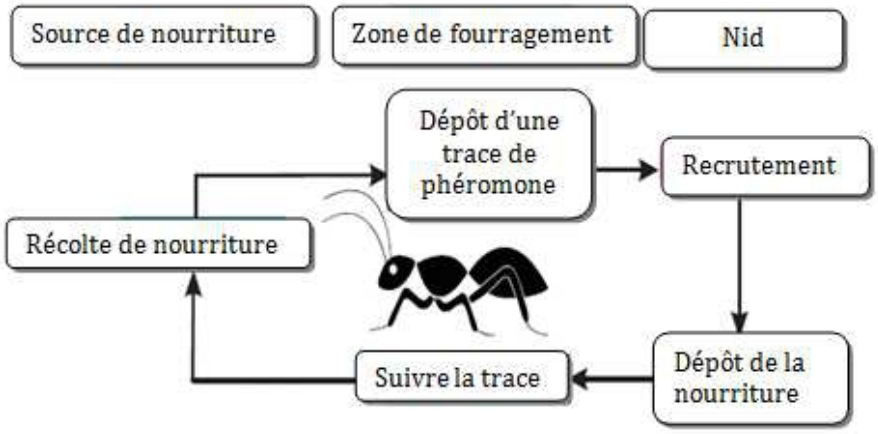


Fig.3.4 Comportement d'une fourmi naturelle lors de la recherche de nourriture

3.4.2 Inspiration Biologique

Les différentes méta-heuristiques ont été inspirées par les travaux des biologistes qui ont longuement été intrigués par le comportement des insectes sociaux en générale (abeilles, termites,…) et les fourmis en particulier (1983 Deneubourg Jean Louis Biologiste, étudie le comportement des Fourmis). Ces derniers sont capables de résoudre collectivement des problèmes complexes, comme trouver le plus court chemin entre une source de nourriture et leur nid par des moyens très simples dans un environnement accidenté. Pour cela, elles communiquent entre elles de façon locale et indirecte, grâce à une hormone volatile, appelée **Phéromone** : Au cours de sa progression, une fourmi laisse derrière elle une trace de phéromone (une sorte de marquage du chemin, elle fournit une information sur la qualité des chemins empruntés afin d'attirer et guider les fourmis) [Dréo, 2003]. Ce procédé basé sur le mécanisme de rétroaction positive, assure que pendant le fourragement pour la nourriture, les fourmis utilisent la voie d'accès la plus courte car elle sera le plus imprégnée par la phéromone (voir figure **Fig.3.4**) [Ouadfel, 2006].

3.4.2.1 Expériences :

Comment peut-on comprendre que les fourmis trouvent le plus cours chemin?

C'est un problème global, malgré que les fourmis n'aient qu'un savoir local [Ouadfel, 2006].

3.4.2.1.1 Pont binaire de Deneubourg :

L'expérience montre un nid d'une colonie de fourmis, qui est séparé d'une source de nourriture par un pont à deux voies de même longueur. On laisse évoluer les fourmis sur le pont, on trace ainsi en fonction du temps, le graphe du nombre de

fourmis empruntant chaque branche. Le résultat de l'expérience est exposé à la figure : **Fig. 3.5.**

L'illustration (a) représente la configuration physique de l'expérience. Le graphique (b) indique l'évolution de ce système en fonction du temps : on constate que les fourmis ont tendance à emprunter le même chemin (par exemple celui du haut) après une dizaine de minutes.

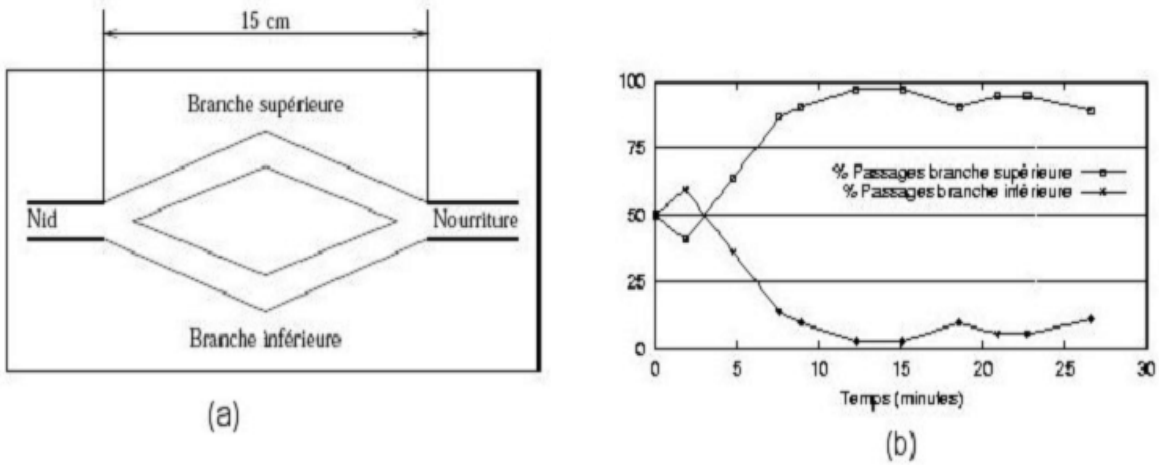


Fig.3.5 Expérience de pont binaire

3.4.2.1.2 Explication :

Au départ, il n'y a pas de phéromone sur le pont. Donc, chaque branche peut être choisie par une fourmi avec la même probabilité. Néanmoins, dans notre exemple, après une certaine période, des variations aléatoires ont fait qu'un peu plus de fourmis ont choisi le chemin du haut plutôt que celui du bas.

Puisque les fourmis déposent des phéromones en avançant et puisqu'elles sont plus nombreuses en haut qu'en bas, le chemin du haut comportera plus de phéromones. Cette quantité supérieure de phéromone incite plus de fourmis à choisir la branche du haut, donc la quantité de phéromone déposée augmentera encore plus.

On en déduit que plus les fourmis ne suivent un chemin, plus ce chemin devient intéressant à suivre. Ainsi la probabilité avec laquelle une fourmi choisit un chemin, augmente avec le nombre de fourmis qui ont pris ce chemin précédemment.

3.4.2.1.3 Expérience du double pont binaire :

On peut se demander à présent quel serait l'effet de l'augmentation de la longueur d'une des deux branches du pont. L' effet produit sera que la branche la plus courte sera sélectionnée.

En effet, les premières fourmis qui reviennent au nid avec de la nourriture sont celles qui ont emprunté le chemin le plus court dans les deux sens. Ce chemin, marqué deux fois par les phéromones, attire plus les autres fourmis que le long chemin, qui lui est marqué une seule fois dans le sens de l'aller. Cet effet se renforce au fur du temps, jusqu'à ce que toutes les fourmis choisissent le chemin le plus court.

C'est ainsi que dans cette expérience, on voit que les variations aléatoires sont réduites, puisque les deux chemins n'ont plus la même longueur. Contrairement à la première expérience, le comportement des fourmis qui consistait à suivre les pistes

de phéromones n'est plus le seul mécanisme présent : maintenant on associe ce mécanisme à une notion de distance.

Toutefois, quand le chemin le plus court n'est ouvert qu'après le chemin plus long (par exemple, quand le chemin était au départ bloqué par un obstacle), les fourmis continuent de parcourir le chemin le plus long car c'est le seul à être recouvert de phéromone. C'est ainsi que, l'évaporation des phéromones joue un rôle capital : les pistes de phéromones sont moins présentes sur les chemins les plus longs, car le réapprovisionnement en phéromone demande plus de temps que sur les chemins plus courts. Donc, il sera alors surtout qu'une poignée de fourmis choisissent le chemin auparavant bloqué pour favoriser celui-ci. Ainsi, même quand des voies plus courtes ont été découvertes après, les fourmis finissent par les emprunter.

On peut généraliser cela à plus de deux chemins possibles (dans la figure : **Fig.3.6**), (a) on a utilisé un double pont avec quatre chemins possibles de différentes longueurs. On voit dans le dessin (b) que la plupart des fourmis finissent par choisir le chemin le plus court. Les expériences montrent que quand environ cent fourmis ont déjà emprunté le pont, plus de 90 pourcents d'entre elles sélectionnent le chemin le plus court : les fourmis convergent donc assez rapidement.

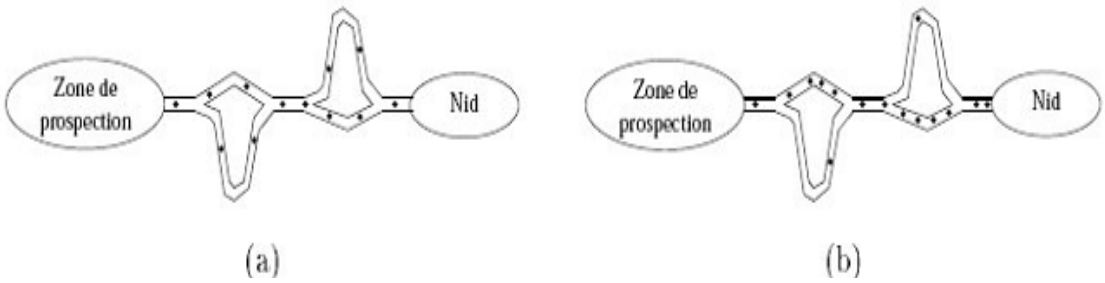


Fig. 3.6. Pont binaire de Deneubourg

3.4.2.1.4 Effet de la coupure d'une piste de phéromone :

Cette fois, les fourmis sont en train de suivre une piste de phéromones, comme présenté à la figure (Fig.3.7), (1) à un moment donné, on a un obstacle qui barre la route des fourmis. Les fourmis qui arrivent à côté de l'obstacle doivent choisir soit d'aller à gauche soit d'aller à droite, (2) Puisqu'aucune phéromone n'est déposée le long de l'obstacle, il y a autant de fourmis qui partent à gauche qu'à droite.

Néanmoins, puisque le chemin de droite est plus court que celui de gauche, les fourmis qui l'empruntent, vont retrouver plus vite la piste de phéromone de départ. Pour chaque fourmi allant du nid à la nourriture, on associe également une fourmi qui va de la nourriture au nid (en fait elles ont été séparées par l'apparition brutale de l'obstacle). Les phéromones de ces fourmis vont se superposer à droite. Donc quand elles vont rejoindre le chemin initial, le chemin de droite sera deux fois plus imprégnée de phéromone que la piste de gauche, où les deux fourmis n'ont pas encore pu rejoindre la piste initiale (ce chemin étant plus long).

Les fourmis qui arrivent à l'obstacle à partir de ce moment, préféreront suivre la piste de droite.

Le nombre de fourmis qui passent par la droite va augmenter, ce qui augmentera encore la concentration de phéromones. De plus, l'évaporation des phéromones sera plus forte sur la piste de gauche du fait que sa longueur est supérieure. La piste de gauche sera dont rapidement abandonnée, parce qu'elle en est beaucoup moins imprégnée : les fourmis passeront toutes très rapidement par la piste la plus courte (3). [Dréo, 2003]

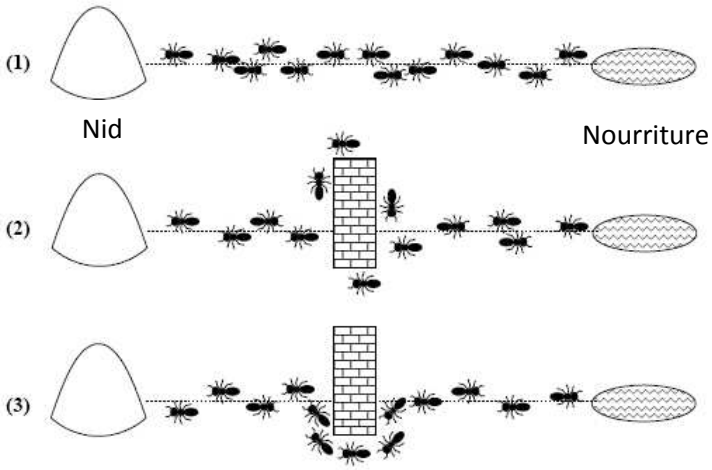


Fig. 3.7 Faculté d'une colonie de fourmis de retrouver le plus court chemin

Il s'agit donc d'une boucle à influence positive sur elle-même où la probabilité qu'une fourmi choisisse un chemin accroît avec le nombre de fourmis qui ont choisi précédemment ce chemin. Ce phénomène est appelé *autocatalytique* [Dréo, 2003].

3.4.2.2 Modèles de fourragement pour la résolution des problèmes d'optimisation

Se basant sur les travaux de Deneubourg, Marco Dorigo et ses collègues furent les premiers à modéliser le comportement de fourragment et à l'appliquer pour résoudre un problème d'optimisation classique: le problème du voyageur du commerce [Dorigo, 1991][Colorni, 1992][Dorigo, 1992]. Cette première modélisation a donné naissance à de nouveaux algorithmes appliqués à d'autres types de problèmes pour lesquels il est difficile de trouver une solution exacte. Cette section présente les différents algorithmes inspirés du comportement des stratégies de recherche de nourriture des fourmis réelles pour résoudre divers problèmes d'optimisation.

3.4.3 Ant System (AS Élitisme)

Est une première variante du "Système de Fourmis". A été proposée par [Dorigo 1996][Costanzo, 2006] : elle est caractérisée par l'introduction de fourmis élitistes. Dans cette version, la meilleure fourmi (celle qui a effectué le trajet le plus court) dépose une quantité de phéromone plus grande, dans l'optique d'accroître la probabilité des autres fourmis d'explorer la solution la plus prometteuse.
AS Elitisme consiste à renforcer de manière artificielle à chaque étape les quantités de phéromone présentes sur les arcs appartenant au meilleur tour trouvé jusqu'à présent.
Intuitivement, cette méthode consiste à faire reparcourir le meilleur tour par certaines fourmis artificielles, dites fourmis élitistes. Ces fourmis sont conceptuellement identiques aux autres fourmis, si ce n'est qu'elles choisissent leur chemin de manière déterministe et qu'elles ne se mettent en route que quand toutes les autres fourmis ont terminé un tour.

$$p_{ij}^{k} = \begin{cases} \dfrac{(\tau_{ij})^{\alpha}(\eta_{ij})^{\beta}}{\sum_{j \in L_k(i)} \left((\tau_{il})^{\alpha}(\eta_{il})^{\beta}\right)} & \text{si } j \in L_k(i) \\ 0 & \text{sinon} \end{cases} \qquad \text{(Eq.3.3)}$$

$L_k(i)$: Est la liste Tabou pour la fourmi (i).

η_{ij} Désigne l'inverse de la distance entre les villes i et j.
α et β Pondèrent l'influence de la phéromone et de la longueur.
τ_{ij} Taux de phéromone entre les villes i et j.

Si l'on désigne par T^* le meilleur tour et par L^* sa longueur, à chaque fin de cycle on réalise l'opération suivante :

$$\tau_{ij(t)} \leftarrow \tau_{ij}(t) + \begin{cases} e.\dfrac{Q}{L^*} & \text{si (i, j)} \in T^* \\ 0 & \text{Sinon} \end{cases} \qquad \text{(Eq.3.4)}$$

où ***e*** est le nombre de fourmis élitistes.

L'effet des fourmis élitistes est de grandement accroître la convergence de la méthode, au risque que celle-ci converge vers une solution non-optimale. Cette amélioration de la convergence s'explique très bien intuitivement, puisque cela revient à favoriser un chemin parmi ceux déjà explorés. Elle est précieuse dans l'AS qui reste un algorithme très gourmand en ressources de calcul.
Il y a donc une valeur optimale à ***e***. Quand nous sommes sous cette valeur et que nous augmentons e, le meilleur tour peut être découvert plus tôt. Quand nous sommes au-delà de cette valeur, les fourmis élitistes forcent l'exploration autour de circuits non-optimaux dès les premières étapes de la recherche, ce qui amoindrit les performances.
Notons aussi que la présence de fourmis élitistes peut même permettre l'émergence de meilleurs tours. En effet, les fourmis élitistes sont un mécanisme qui permet de focaliser la recherche sur des zones prometteuses de l'espace de recherche en augmentant l'attrait des fourmis pour des pistes que l'on sait faire partie d'un bon tour. Or, quand un bon tour a été découvert, il y a de forte chance que le tour optimal n'en est pas très éloigné.

3.4.3.1 Résolution par le Ant System (AS) : le problème du voyageur de commerce PVC (TSP : Travelling Salesman Problem) [Dubuisson, 2006]

Le problème du voyageur de commerce ou PVC consiste à trouver le plus court chemin permettant de relier un ensemble de villes en ne passant qu'une et une seule fois par une ville.
Le problème revient à trouver le cycle hamiltonien minimal dans un graphe complet pondéré où les sommets sont les villes et les arcs sont les chemins entre les villes.

Le TSP a fait l'objet de la première implémentation d'un algorithme de colonies de fourmis : le Ant System (AS). Nous allons maintenant décrire cet algorithme.
Soit le graphe complet pondéré $G = (V;E)$.
À chaque itération t ($1 \leq t \leq t_{max}$) chaque fourmi k ($k = 1,\ldots,m$) parcourt le graphe et construit un trajet complet de $n = |V|$ étapes ($|V|$ étant le cardinal de l'ensemble de sommets V). Pour chaque fourmi, le trajet entre la ville i et la ville j dépend de :

1. La liste des villes que la fourmi *k* peut visité quand elle est sur la ville *i* (c'est-à-dire les villes qui sont reliées à *i* par une arête et que la fourmi *k* n'a pas encore visitée) : J_i^k
2. L'inverse de la distance entre les villes : $\eta_{ij} = \frac{1}{d_{ij}}$, appelée visibilité ou valeur heuristique
3. La quantité de phéromones déposée sur l'arête reliant les deux villes : τ_{ij}

On utilise des lois de transition aléatoires où $p_{ij}^k(t)$ est la probabilité que la fourmi *k*, situé sur la ville *i* à l'itération *t* se déplace vers la ville *j*. La probabilité $p_{ij}^k(t)$ est donnée par la formule :

$$p_{ij}^k(t) = \begin{cases} \dfrac{(\tau_{ij}(t))^\alpha (\eta_{ij}(t))^\beta}{\sum_{l \in J_i^k} \left((\tau_{il}(t))^\alpha (\eta_{il}(t))^\beta\right)} & \text{si } j \in J_i^k \\ 0 & \text{si } j \notin J_i^k \end{cases} \qquad \text{(Eq.3.5)}$$

Où α et β sont deux paramètres à ajuster qui contrôlent l'importance relative donnée aux traces de phéromones $\tau_{ij}(t)$ par rapport à l'heuristique η_{ij}.
Avec $\alpha=0$, seule la visibilité de la ville est prise en compte. Inversement, avec $\beta=0$, seules les pistes de phéromones influent. Pour éviter une sélection trop rapide d'un trajet, un compromis entre ces deux paramètres doit être trouvé. Cela joue sur les comportements de ***diversification*** et ***intensification***.
Après un tour complet, chaque fourmi laisse une certaine quantité de phéromones $\Delta\tau_{ij}^k(t)$ sur l'ensemble de son parcours. Cette quantité dépend de la qualité de la solution trouvée :

$$\Delta\tau_{ij}^k(t) = \begin{cases} \dfrac{Q}{L^k(t)} & si\ (i,j) \in T^k(t) \\ 0 & si\ (i,j) \notin T^k(t) \end{cases} \qquad \text{(Eq.3.6)}$$

Où $T^k(t)$ est le trajet effectué par la fourmi k à l'itération *t*, $L^k(t)$ la longueur du tour et Q un paramètre fixé.
Enfin, le processus d'*évaporation* entre en jeu. Cela permet d'éviter d'être piégé dans des solutions sous-optimales en faisant « oublier » au système les mauvaises solutions. A chaque itération, la règle de mise-à-jour est la suivante :

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \Delta\tau_{ij}(t) \qquad \text{(Eq.3.7)}$$

Où $\Delta\tau_{ij}(t) = \sum_{k=1}^{m} \Delta\tau_{ij}^k(t)$ avec *m* qui est le nombre de fourmis, et $\rho \in [0;1]$ paramètre d'évaporation.
La quantité initiale de phéromones sur chaque arête est une distribution uniforme d'une petite quantité $\rho \geq 0$.
Enfin, nous pouvons conclure cette partie en proposant un algorithme en pseudo-code issu de [Dréo, 2003].

Algorithme 3.1 Algorithme Ant System

Initialisation
Pour $t=1,\ldots,t_{max}$
  Pour chaque fourmi $k=1,\ldots,m$
    Choisir une ville au hasard
    Pour chaque ville non visitée i
      Choisir une ville j, dans la liste J_i^k des villes restantes, selon la formule (Eq.3.5)
    Fin Pour
    Déposer une piste $\Delta\tau_{ij}^k(t)$ sur le trajet $T^k(t)$ conformément à l'équation (Eq.3.6)
  Fin Pour
  Evaporer les pistes selon la formule (Eq.3.7)
Fin Pour

3.4.3.2 Fonctionnement de l'algorithme [Costanzo, 2006]

Nous allons maintenant préciser les différentes étapes de l'algorithme au cours du temps.

Initialisation de l'algorithme. Les éléments s'agencent de la manière suivante au début de l'algorithme :

1. Les m fourmis sont réparties aléatoirement sur les n villes.
2. Pour chaque fourmi, la liste qui modélise sa mémoire contient sa ville de départ.
3. Les pistes de phéromones sont initialisées comme suit : $t_{ij}(0) = c$, où c est une petite constante positive, qui ne peut être nulle (sinon, il y a un problème lors du calcul de (Eq.3.5))

Fin d'un cycle. Après n itérations, nous sommes à l'instant t, toutes les fourmis ont terminé leur tour, chacune a une liste "mémoire" pleine et est revenue à sa propre ville de départ. À ce moment :

1. Chaque fourmi calcule sa valeur $L_k(t)$.
2. Le calcul des variables $\tau_{ij}^k(t)$.
3. Les variables de phéromone $\tau_{ij}(t)$ sont mises à jour suivant la formule (Eq.3.7). En d'autres termes, la fourmi refait son tour en sens inverse tout en déposant des phéromones.
4. On observe quelle fourmi a trouvé le tour de longueur minimum (i.e. on recherche la fourmi k telle que $k = \min_{k=1}^{m} L_k(t)$. Si ce tour est meilleur que le meilleur tour jusqu'ici, on le mémorise.
5. Les mémoires des fourmis (liste des villes visitées) sont effacées.
6. Les fourmis recommencent un nouveau tour, toujours au départ de la ville sur laquelle elles avaient été placées au début de l'algorithme.

Fin de l'algorithme. On arrête l'algorithme après un nombre de cycles égal à une constante NC_{max}. Si à partir d'un instant, toutes les fourmis font le même tour, l'algorithme s'interrompt : on est dans une situation de stagnation où le programme arrête de chercher des alternatives. L'algorithme donne en retour le meilleur tour mémorisé.

Les paramètres permettant de caractériser complètement une instance de AS sont repris dans les tuplet : $< m, \rho, \alpha, \beta, Q, NC_{max} >$,où $0 \leq \rho < 1$, $\alpha \geq 0$, $\beta \geq 0$ et $c > 0$.

3.4.3.3 Complexité de l'algorithme AS

Afin de se faire une idée, nous divisons l'algorithme en 5 sections pour calculer la complexité [Dréo 04] :

1. Initialisation.
2. Un cycle de l'algorithme.
3. Fin du cycle et calcul des dépôts de phéromone.
4. Evaporation des phéromones.
5. Boucle de l'algorithme.

Procédons étape par étape :

1. On note L l'ensemble des chemins du graphe. Comme les sommets (villes) sont connexes deux à deux, le cardinal de L vaut $|L|=n(n-1)/2=O(n^2)$
   On a une complexité $O(|L| + m) = O(n^2 + m)$, puisque l'on a supposé une interconnexion totale entre les villes puis l'ensemble des fourmis.
2. La complexité est $O(n^2 \cdot m)$, puisque les opérations de calcul de la ville suivante nécessite un balayage de l'intégralité des villes.
3. La complexité est $O(|L| + m \cdot |L|) = O(m \cdot |L|) = O(n^2 \cdot m)$.
4. La complexité est $O(|L|) = O(n^2)$.
5. Le test de stagnation est de complexité $O(n \cdot m)$ (on doit comparer les tours de m fourmis, chaque tour ayant une longueur de n éléments).

La complexité globale est obtenue en additionnant la complexité de l'étape 1, au produit du nombre total de cycle (soit NC_{max}) par la complexité globale des étapes 2 à 5. La complexité globale étant celle maximale, soit $O(n^2*m)$. La complexité générale de l'algorithme devient donc : $\mathbf{O(n^2 + m + NC_{max} \cdot n^2 \cdot m)}$
soit : $\mathbf{AScomplexity = O(NC_{max} \cdot n^2 \cdot m)}$
Il faut noter cependant que cette formule ne nous dit rien sur le nombre d'étapes qui sont effectivement nécessaires pour atteindre l'optimum : on pourrait atteindre celui-ci bien avant les NC_{max} cycles.

Exemple : la complexité de AS Elitiste : Le NC_{max} est fixé en fonction des ressources de calcul qui sont allouées à la résolution du problème.
Rien n'interdit d'interrompre la recherche selon un critère probabiliste sur la distribution de la population des $L_k(t)$. Puisque m = n (nombre de ville est égal au nombre de fourmis), la complexité au pire des cas de l'algorithme est donnée par la formule $\mathbf{O(NC_{max} \cdot n^2 \cdot m)}$ devient : $\mathbf{AScomplexity = O(NC_{max} \cdot n^3)}$
Au final, AS est un algorithme cubique, car $\mathbf{NC_{max}}$ est une constante.

3.4.4 Ant Colony System « ACS »

L'algorithme « Ant Colony System » a été introduit par « Dorigo » et Grambardella » en 1996 pour améliorer la performance de AS [Dorigo, 1997b].
L'algorithme a été introduit pour améliorer les performances du premier algorithme sur des problèmes de grandes tailles [Dorigo, 1997a] [Dorigo, 1997b]. ACS est fondé sur des modifications de l'AS :

1. ACS introduit une règle de transition dépendant d'un paramètre q_0 $(0 \le q_0 \le 1)$, qui définit une balance diversification/intensification. Une fourmi k sur une ville i choisira une ville j par la règle :

$$j = \begin{cases} \arg \max_{J} \left([\tau_{ij}]^{\alpha} [\eta_{ij}]^{\beta} \right) & \text{si } q \le q_0 \\ J & \text{sinon} \end{cases} \qquad \text{(Eq.3.8)}$$

Et $J \in L_{K(i)}$ une ville sélectionnée aléatoirement selon la probabilité :

En fonction du paramètre q_0, il y a donc deux comportements possibles :
Si $q>q_0$ le choix se fait de la même façon que pour l'algorithme AS, et le système tend à effectuer une diversification ; si q et q_0, le système tend au contraire vers une intensification. En effet, pour q et q_0, l'algorithme exploite davantage l'information récoltée par le système, il ne peut pas choisir un trajet non exploré.

2. La gestion des pistes est séparée en deux niveaux : une mise à jour locale et une mise à jour globale. Chaque fourmi dépose une piste lors de la mise à jour locale :

$$\tau(t+1) = (1-\rho)\tau(t) + \rho\tau_0 \qquad \text{(Eq.3.9)}$$

Où τ_0 est la valeur initiale de la piste. A chaque passage, les arêtes visitées voient leur quantité de phéromone diminuer, ce qui favorise la diversification par la prise en compte des trajets non explorés. A chaque itération, la mise à jour globale s'effectue comme ceci :

$$\tau(t+1) = (1-\rho)\tau(t) + \rho\Delta(t) \qquad \text{(Eq.3.10)}$$

Ici, seule la meilleure piste est donc mise à jour, ce qui participe à une intensification par sélection de la meilleure solution

3.4.5 Ant-Q

Dans cette variante de AS, la règle de mise à jour locale est inspirée du **Q-learning** [Dorigo, 1995]. Cependant, aucune amélioration par rapport à l'algorithme AS n'a pu être démontrée. Cet algorithme n'est d'ailleurs, de l'aveu même des auteurs, qu'une préversion du "Ant Colony System"

$$j = \begin{cases} \arg \max_{J} \left([\tau_{ij}]^{\alpha} [\eta_{ij}]^{\beta} \right) & \text{si } q \le q_0 \\ J & \text{sinon} \end{cases} \qquad \text{(Eq.3.11)}$$

$\boldsymbol{\eta_{ij}}$ Est une valeur fournie par une heuristique.
$\boldsymbol{\tau_{ij}}$ Donne la valeur de probabilité de choix (phéromone).
$\boldsymbol{\alpha}$ et $\boldsymbol{\beta}$ Pondèrent l'influence des deux mesures.
$\boldsymbol{q_0}$ Est la probabilité d'utiliser la première équation.

3.4.6 Max-Min Ant System

Dans [Stützle, 1999] [Taillard, 1999] Stutzle et Hoos introduisent l'algorithme nommé MMAS. Les modifications introduites concernent :

L'utilisation de deux constantes $\boldsymbol{\tau_{min}}$ et $\boldsymbol{\tau_{max}}$ comme borne inférieure et supérieure à la quantité de phéromone présentent sur les arcs du graphe. Ces deux valeurs permettent de limiter les variations des taux de phéromone et éviter ainsi que certains arcs soient totalement délaissés au profit d'autres ce qui est reconnu comme un état de stagnation prématurée. Toutes les traces de phéromones sont initialisées à $\boldsymbol{\tau_{max}}$. Les quantités de phéromones sont initialisées à la valeur maximale ; La mise à jour des traces de phéromones n'est autorisée que par la fourmi ayant trouvé la meilleure solution.

3.4.7 L'heuristique ACO

Toutes les variantes que nous venons d'exposer ont été regroupées sous une description plus large : l'heuristique ***ACO*** (Ant Colony Optimization), afin de faciliter le rapprochement des méthodes entre elles et de se soustraire aux spécificités du PVC [Stützle, 1999b] [Dorigo, 1999a][Dorigo, 1999c]. Dans cet effort de généralisation, on peut

noter l'introduction (hasardeuse) d'un processus « reine » visant à coordonner et superviser le travail des fourmis [Taillard, 1999].

3.4.8 Autres domaines d'application

Les bonnes performances des algorithmes basés fourmis obtenues lors de leur application au problème du voyageur du commerce ont incité beaucoup de chercheurs à les utiliser dans d'autres domaines d'application. Sans vouloir dresser une liste exhaustive de toutes les applications et variantes qui ont été produites, on peut citer le problème d'affectation quadratique [Taillard, 1999] [Maniezzo, 1994], le problème de coloration de graphe [Costa, 1997] [Dudot, 2005], le problème de routage [Helton, 2005], les réseaux de communication et le problème d'ordonnancement [Gambardella, 1997] [Merkel, 2002], les problèmes de satisfaction de contraintes [Solnon, 2000], la fouille de données [Parpinelli, 2002], l'optimisation de site d'enseignement en ligne (notion d' « E-Learning » [Semet, 2003]

3.4.9 Formalisation et propriétés d'un algorithme de colonie de fourmis [Dréo, 2004]

Une description élégante a été proposée [Dorigo, 2003], qui s'applique aux problèmes (combinatoires) où une construction partielle de la solution est possible. Ce cas, bien que restrictif, permet de dégager les apports originaux de ces méta-heuristiques (dénommées *ACO*, pour « Ant Colony Optimization »). Nous en avons traduit ci-dessous un extrait : Une méta-heuristique de colonie de fourmis est un processus stochastique construisant une solution, en ajoutant des composants aux solutions partielles.

Ce processus prend en compte (i) une heuristique sur l'instance du problème (ii) des pistes de phéromone changeant dynamiquement pour refléter l'expérience acquise par les agents.

Une formalisation plus précise existe [Dorigo, 2003]. Elle passe par une représentation du problème, un comportement de base des fourmis et une organisation générale de la méta-heuristique. Plusieurs concepts sont également à mettre en valeur pour comprendre les principes de ces algorithmes, notamment la définition des pistes de phéromone en tant que mémoire adaptative, la nécessité d'un réglage intensification/diversification et enfin l'utilisation d'une recherche locale. Nous traitons ci-après ces différents sujets.

3.4.9.1 Formalisation :

- **Représentation du problème**. Le problème est représenté par un jeu de solutions, une fonction objective assignant une valeur à chaque solution et un jeu de contraintes. L'objectif est de trouver l'optimum global de la fonction objectif satisfaisant les contraintes. Les différents états du problème sont caractérisés comme une séquence de composants. On peut noter que, dans certains cas, un coût peut être associé à des états autres que des solutions **[Dréo, 2004]**.

(a) **(b)**

Fig.3.8 Association des pistes de phéromones aux composants (a) ou aux connexions (b) du graphe représentant le problème à résoudre.

Dans cette représentation, les fourmis construisent des solutions en se déplaçant sur un graphe $G = (C; L)$, où les nœuds sont les composants de C et où l'ensemble L connecte les composants de C. Les contraintes du problème sont implémentées directement dans les règles de déplacement des fourmis (soit en empêchant les mouvements qui violent les contraintes, soit en pénalisant de telles solutions).

- **Comportement des fourmis.** Les fourmis artificielles peuvent être caractérisées comme une procédure de construction stochastique construisant des solutions sur le graphe G = (C; L). En général, les fourmis tentent d'élaborer des solutions faisables mais, si nécessaire, elles peuvent produire des solutions infaisables. Les composants et les connexions peuvent être associés à des pistes de phéromone τ (mettant en place une mémoire adaptative décrivant l'état du système) et à une valeur heuristique η (représentant une information a priori sur le problème, ou venant d'une source autre que celle des fourmis ; c'est bien souvent le coût de l'état en cours). Les pistes de phéromone et la valeur de l'heuristique peuvent être associées soit aux composants, soit aux connexions (figure **Fig.3.8**) [Dréo, 2004].
  Chaque fourmi dispose d'une mémoire utilisée pour stocker le trajet effectué, d'un état initial et de conditions d'arrêt. Les fourmis se déplacent d'après une règle de décision probabiliste fonction des pistes de phéromone locales, de l'état de la fourmi et des contraintes du problème. Lors de l'ajout d'un composant à la solution en cours, les fourmis peuvent mettre à jour la piste associée au composant ou à la connexion correspondante. Une fois la solution construite, elles peuvent mettre à jour la piste de phéromone des composants ou des connexions utilisées. Enfin, une fourmi dispose au minimum de la capacité de construire une solution du problème.
- **Organisation de la méta-heuristique**. En plus des règles régissant le comportement des fourmis, un autre processus majeur a cours : l'évaporation des pistes de phéromone. En effet, à chaque itération, la valeur des pistes de phéromone est diminuée. Le but de cette diminution est d'éviter une convergence trop rapide et le piégeage de l'algorithme dans des minimums locaux, par une forme d'oubli favorisant l'exploration de nouvelles régions [Dréo, 2004].

3.4.9.2 Phéromones et mémoire

L'utilisation de la stigmergie est cruciale pour les algorithmes de colonies de fourmis. Le choix de la méthode d'implémentation des pistes de phéromone est donc important pour obtenir les meilleurs résultats. Ce choix est en grande partie lié aux possibilités de représentation de l'espace de recherche, chaque représentation pouvant apporter une façon différente d'implémenter les pistes. Par exemple, pour le problème du voyageur de commerce, une implémentation efficace consiste à utiliser une piste τ_{ij} entre deux villes i et j comme une représentation de l'intérêt de visiter la ville j après la ville i [Dréo, 2004].

En effet, les pistes de phéromone décrivent à chaque pas l'état de la recherche de la solution par le système, les agents modifient la façon dont le problème va être représenté et perçu par les autres agents. Cette information est partagée par le biais des modifications de l'environnement des fourmis, grâce à une forme de communication indirecte : la stigmergie.

L'information est donc stockée un certain temps dans le système, ce qui a amené certains auteurs à considérer ce processus comme une forme de mémoire adaptative

[Taillard, 1998], où la dynamique de stockage et de partage de l'information va être cruciale pour le système.

3.5 Conclusion

L'objectif de ce chapitre était de présenter un aperçu sur l'optimisation et ses techniques d'un coté et de se fonder sur celle des fourmis par leurs outils (algorithmes) de l'autre coté.

On a commencé par une définition de l'optimisation et ses composantes. On a opté pour l'optimisation combinatoire ainsi de l'optimisation difficile. Du moment que les algorithmes de colonies de fourmis sont l'objectif de ce chapitre, on les a détaillés avec la spécification de sa complexité algorithmique. On a terminé ce chapitre par une formalisation et des propriétés des algorithmes de colonies de fourmis.

Chapitre 4

Le problème de la classification non supervisée

Chapitre 4 : Le problème de la classification non supervisée

4.1 Introduction

Il existe plusieurs définitions de la classification, voici quelques une :

- Le but des méthodes de classification est de construire une partition ou une suite de partitions emboîtées, d'un ensemble d'objets dont on connaît les distances deux à deux. Les classes formées doivent être le plus homogène possible » **[Saporta, 1990]**

- La classification est l'organisation d'un ensemble de données en classes homogènes. Elle a pour but de simplifier la représentation des données initiales. La classification automatique, appelée également classification non-supervisée (Clustering), recouvre l'ensemble des méthodes permettant la construction automatique de telles classification. **[Charles, 2004]**

- La classification automatique cherche à trouver une partition de l'espace de départ tel que les données appartenant à un même groupe soient le plus similaire entre elles qu'avec les données issues d'un autre groupe. **[Labroche, 2003]**

Le processus de classification peut être résumé par le fait de grouper des objets ensemble selon des critères. Quand nous désignons, par exemple, d'un coté les êtres humains et de l'autre les animaux, nous effectuons une classification, et ceci de manière tout à fait naturelle. La classification automatique est la reproduction via une machine de ce processus naturel à l'homme, qui permet de regrouper des objets similaires sans connaissance acquise au préalable.

Vu le nombre de problèmes de classification et de leurs domaines d'application, un grand nombre de méthodes de classification ont vu le jour. Cependant la plupart de ces méthodes partage des caractéristiques communes. Il a donc été possible de les regrouper dans une hiérarchie de méthodes selon: les sorties désirées (ensembles disjoints ou flous, hiérarchie de classe ou partition), appréhension du problème (apprentissage supervisé ou non).

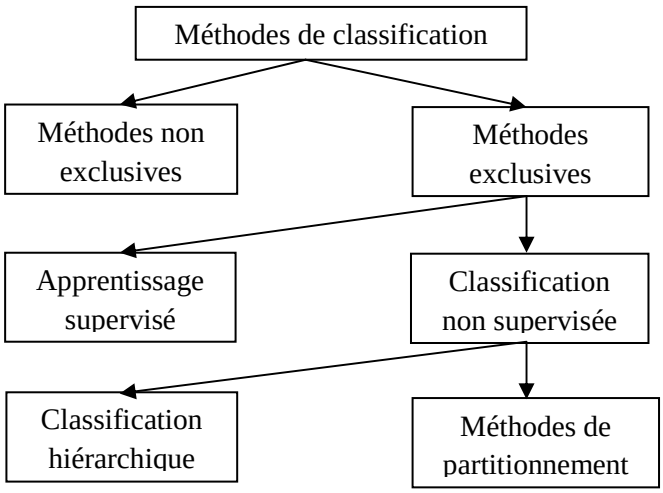


Fig.4.1 Taxinomie de méthodes de classification

Les méthodes de classification sont divisées en deux parties, les méthodes exclusives et les méthodes non exclusives [Fig.4.1]. Ce qui différencie ces deux types de méthodes est l'appartenance des objets aux classes, dans une méthode dite exclusive les objets n'appartiennent qu'à une seule classe dans la partition finale, alors que pour une méthode non exclusive, chaque objet se verra attribuer une probabilité d'appartenance à chaque classe résultante.

La classification peut être approchée selon deux objectifs différents. Tout d'abord elle peut servir une étude plus précise des données traitées, en les partitionnant en groupes plus petits qui pourraient exprimer une information plus claire que celle exprimé par la masse de données. Cette approche est appelé classification non-supervisée dans le sens où elle ne reçoit aucune autre information extérieure que les données en elles mêmes. La deuxième approche, par opposition à la précédente, est appelée classification supervisée dans le sens où l'intervention d'un système extérieur à la méthode est requis afin d'étiqueter les données au préalable et d'effectuer un apprentissage [Fig.4.3] à la méthode afin qu'elle s'entraine à étiqueter les données. L'objectif de cette classification est de concevoir une machine capable d'assigner toute observation inconnue à une classe déjà définie sans intervention du système extérieur qui est constitué le plus souvent d'un opérateur humain expert du domaine d'application de la méthode.

Les approches de classification supervisées ou non supervisées englobent plusieurs méthodes ; nous retrouvons particulièrement les méthodes « classiques » faisant appel à des outils et des théories mathématiques diverses [Duda, 1973][Rich, 1993][Rich, 1998][Belh, 1998]. Ces méthodes sont généralement, soient statistiques basées sur une mesure de similarité probabiliste, telle que, la méthode de maximum de vraisemblance, soient géométriques basées sur une mesure de similarité métrique, telle que, la méthode de minimum de distance euclidienne. La caractéristique principale de ces méthodes est quelles sont non itératives et aboutissent à un seul résultat.

Par ailleurs, il existe une grande famille de méthodes qui énoncent le problème de la classification en termes d'optimisation d'un critère mathématique bien défini. La caractéristique principale de ces méthodes itératives, appelées « heuristiques », est la problématique nouvelle qu'elles ont introduite. Il s'agit d'optimiser un critère qui

exprime l'adéquation entre le fait que « les classes distinctes doivent être séparables et les objets similaires doivent être groupés dans une même classe » [Khedam, 2008].

Le problème d'optimisation se pose alors, en terme de « recherche simultanée » de la classification et de la représentation des classes de cette classification parmi un ensemble de classifications et de représentations possibles, qui optimisent le critère prédefini » [Cell, 1989]. Ce critère mathématique est formalisé par une fonction, dite « fonction objectif » ou « fonction coût ». Pour chaque solution, elle renvoie une valeur, et la solution optimale (meilleure solution) est celle qui minimise (ou maximise) la fonction objectif. Trouver cette solution optimale dans un ensemble discret et fini est un problème facile en théorie : il suffit d'essayer toutes les solutions, et de comparer leurs qualités pour voir la meilleure. Cependant, en pratique, l'énumération de toutes les solutions peut prendre un temps assez long. De ce fait, dans la plupart des cas, on se contente d'avoir une solution approchée, obtenue par une heuristique.

Les heuristiques sont des algorithmes stochastiques itératifs, qui progressent vers l'optimum par échantillonnage de la fonction objectif. Elles se comportent comme des algorithmes de recherche, tentant d'apprendre les caractéristiques du problème afin d'en trouver une approximation de la meilleure solution. Il existe un grand nombre d'heuristiques différentes, allant de la simple recherche locale à des algorithmes complexes de recherche globale. Ces méthodes utilisent cependant un haut niveau d'abstraction, leur permettant d'être adaptées à une large gamme de problèmes différents, d'où l'origine de leur appellation par « méta-heuristiques » [Dreo, 2003]. La méthode des nuées dynamiques et la méthode ISODATA font partie des premières méthodes de classification heuristiques qui ont suscité un grand intérêt et une large utilisation dès leur apparition dans les années 80 [Cell, 1989]. Cependant, de nouvelles heuristiques ont émergé. Elles sont, pour la plupart, inspirées des systèmes classifieurs naturels, qu'ils soient pris en physique, telle que la méthode du « recuit simulé », ou en biologie de l'évolution, tels que les algorithmes génétiques, ou en biologie du comportement (éthologie), tels que les algorithmes de colonies de fourmis ou l'optimisation par essaims particulaires. Ces nouvelles heuristiques inspirées de la biologie constituent à part entière une catégorie de méthodes de classification que l'on peut nommer « les heuristiques biomimétiques » [Monm, 2000][Azza, 2004].

Dans notre travail de recherche, nous nous intéressons aux approches de classification dont le cadre théorique permet la conception de modèles décisionnels capables de réaliser une opération de classification.

4.2 La biomimétique et la classification [Pierrick, 2010]

Le problème de l'organisation et de la classification se rencontre souvent dans la nature. Les algorithmes évolutionnaires EAs (Evolutionary Algorithms) inspirés de la génétique des populations [Holl, 1975], ont déjà donné lieu à de nombreux travaux [Cucc, 1993], [Deb, 1998], [Gold, 1989] [Mich, 1996], traitant de la classification des données en général, et des images en particulier. Aussi, de nouveaux classifieurs biomimétiques ont émergé. Parmi ces derniers, figurent ceux qui comme les algorithmes évolutionnaires sont inspirés des mécanismes biologiques, nous citons par exemple, le système immunitaire artificiel AIS (Artificial Immune System) inspiré du fonctionnement du système immunitaire naturel [DeCa, 2002]. On retrouve également ceux qui sont inspirés de la biologie du comportement (éthologie), tels que l'algorithme des fourmis classifieuses (AntClass) inspirés du comportement intelligent des fourmis réelles [Dori, 2000], et l'algorithme à essaim particulaires PSO (Particle Swarm

Optimizer) inspiré du déplacement collectif et harmonieux de certaines espèces d'oiseaux et de poissons [Kenn, 1995].

Le principal avantage de ces méthodes heuristiques vient de leur capacité à traiter le problème de la classification.

L'objectif principal de notre thèse est de revoir, exploiter et adapter les classifieurs biomimétique orienté fourmis artificielles sur plusieurs types de données.

4.3 Classification des images satellites

Dans ce chapitre on traite la classification des données en choisissant le type de données image satellite, tandis que dans les prochains chapitres on verra l'application des différents algorithmes utilisés sur plusieurs types de données afin de sortir d'une conclusion sur l'effet des données sur les différents algorithmes utilisés et vice versa.

La classification dans l'imagerie satellitaire consiste à identifier des groupes homogènes de pixels qui ont les mêmes similitudes spectrales afin de fournir une information résumée, simplifiée, et facilement interprétables.

Cependant, elle utilise l'information spectrale contenue dans les valeurs d'une ou de plusieurs bandes spectrales pour classifier chaque pixel individuellement [Fig.4.2]. Ce type de classification est appelé reconnaissance de regroupements spectraux. Ils ont pour but d'assigner une classe particulière ou thème (par exemple : eau, forêt de conifères, blé, etc.) à chacun des pixels d'une image. La "nouvelle" image qui représente la classification est composée d'une mosaïque de pixels qui appartiennent chacun à un thème particulier. Cette image est essentiellement une représentation thématique de l'image originale « carte thématique ».

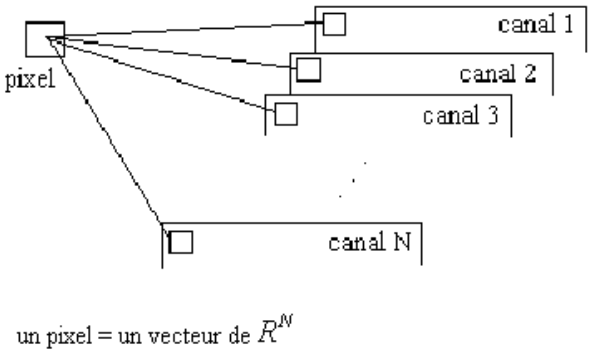


Fig.4.2 Image multispectrale

Lorsqu'on parle de classes, il faut faire la distinction entre des classes d'information et des classes spectrales. Les classes d'information sont des catégories d'intérêt que l'analyste tente d'identifier dans les images, comme différents types de cultures, de forêts ou d'espèce d'arbres, différents types de caractéristiques géologiques ou de roches, etc .

Les classes spectrales sont des groupes de pixels qui ont les mêmes caractéristiques (ou presque) en ce qui a trait à leur valeur d'intensité dans les différentes bandes spectrales des données.

L'objectif ultime de la classification est de faire la correspondance entre les classes spectrales et les classes d'information. L'image constituée représente une bonne source d'information. Il est rare qu'une correspondance directe soit possible entre ces deux types de classes. Des classes spectrales bien définies peuvent apparaître parfois sans

qu'elles correspondent nécessairement à des classes d'information intéressantes pour l'analyse [Jensen, 1983][Emmanuel, 2000].

L'analyste a le rôle de déterminer l'utilité des différentes classes spectrales et de valider leur correspondance à des classes d'informations utiles.

L'objectif de la phase d'apprentissage [fig.4.3], dans le cas supervisée, est de repérer, au sol, des parcelles qui seront aussi représentative que possible des phénomènes à observer et qui servent à établir une relation entre la radiométrie et le contenu de la parcelle.

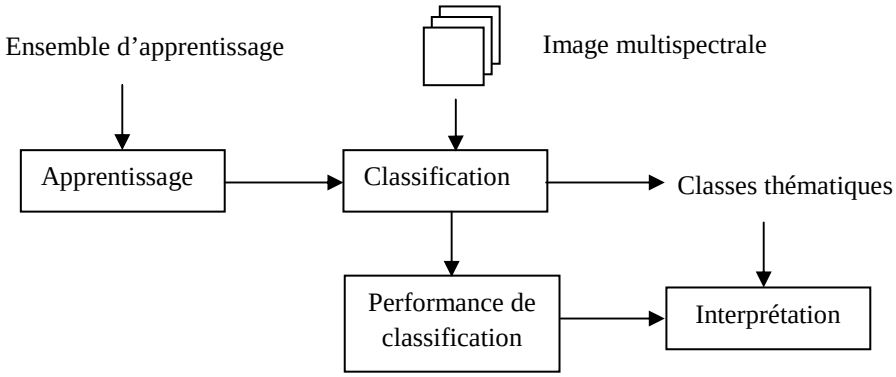


Fig.4.3 Schéma de principe de la classification supervisée

Les approches de partitionnement ou de classification non supervisées sont des techniques relativement primitives, dans la mesure où elles s'effectuent sans aucune connaissance a priori ni sur le nombre de classes ni sur la structure de ces classes [Smara, 1998]. Le groupement des points de l'image « «pixels » est réalisé à partir d'une mesure de similarité d'attributs pour former des classes spectrales. Ces dernières peuvent ne pas correspondre aux classes thématiques réelles. Une interprétation a posteriori de ce résultat par un thématicien (expert) est alors indispensable.

4.4 Techniques de classification

Le choix de la méthode dépend de la nature des données en entrée et de la classification désirée.

Dans le cas de la classification non supervisée, l'objectif est de détecter des groupements ou des classes au sein de la population d'apprentissage sans aucune connaissance à priori sur les objets à classer. Par conséquent, elle nécessite une quantité minimale d'entrée initiale de la part de l'analyste.

Une méthode de classification non supervisée (en anglais, on parle de clustering ou de unsupervised learning) n'utilise que la matrice de dissimilarité. Les classifications non supervisées ou non dirigées s'appuient sur des méthodes statistiques, telles que la méthode k-Moyennes ou l'ISODAT (voir Annexe C)

On applique la classification non supervisée dans le domaine de la télédétection lorsque l'identité des types de couverture du sol n'est pas connue. Cela résulte d'un manque de l'information ou de l'incertitude sur la réalité du terrain. Il existe des algorithmes de classification, composés de plusieurs itérations, permettant de créer des regroupements de pixels ayant des signatures spectrales similaires. L'utilisateur

procède ensuite à la reconnaissance des classes créées par l'algorithme en affectant un nom et une couleur à chaque classe.

4.5 La classification avec les algorithmes de colonies de fourmis

4.5.1 Travaux fondateurs :

Les fourmis réelles ont inspirés les chercheurs en informatique dans de nombreux domaines. Cela se justifie particulièrement quand on connaît la richesse comportementale de ces animaux.

Trois domaines s'occupent de près à la modélisation des fourmis en terme mathématique et informatique et à l'utilisation concrète de ces modèles.

La première approche s'intéresse à la modélisation des fourmis dans le domaine du tri d'objets artificielles et par la classification.

La deuxième approche développée et appliquée à de nombreux problèmes d'optimisation combinatoire et numérique.

La troisième approche, qui nous intéresse dans notre travail, est l'exploitation du modèle d'optimisation pour résoudre le problème de partitionnement.

Pour la première approche, les algorithmes de classification automatique sont inspirés du comportement de tri collectif observé chez les fourmis. En effet, certains travaux ont montré que certaines espèces de fourmis parviennent à organiser divers éléments du couvain tels que les œufs, les larves,…etc. [Deneubourg, 1990] [Deneubourg, 1991] [Khedam,2008] [Ouadfel, 2006].

Le principe de base de ce comportement est le suivant:

- Lorsqu'une fourmi rencontre un élément du couvain, plus cet élément est isolé, plus elle a de chance de le ramasser;
- Lorsqu'une fourmi transporte un élément du couvain, la probabilité qu'elle le dépose est d'autant plus grande que la densité d'éléments de même type dans le voisinage est grande.

Deneubourg et son équipe [Deneubourg, 1990] furent les premiers à modéliser ce genre de comportement. Lors des expériences de simulation, les objets à rassembler sont placés aussi aléatoirement sur une grille. Les fourmis sont modélisées par de simples agents qui sont placés eux aussi aléatoirement sur la grille représentant l'environnement dans lequel elles évoluent. Chaque agent fourmi n'a qu'une perception locale de son environnement et a pour tâche de déplacer les objets en fonction de la concentration des objets de même type dans leur environnement proche appelé « voisinage ».

Le principe est de regrouper les objets similaires en des groupes sur une grille. Chaque fourmi peut prendre un objet avec une probabilité fonction de sa similarité avec les objets présents dans son voisinage et le déposer selon la même probabilité. Après un certain nombre d'itérations, des groupes d'objets similaires se forment sur la grille. La principale caractéristique de ces algorithmes est leur coté non supervisé qui permet de découvrir automatiquement le nombre de groupe adéquat sans intervention extérieure comme dans les algorithmes classiques de classification. Les opérations de dépôt et de ramassage des objets sont biaisées par les probabilités $\boldsymbol{P_p}$ et $\boldsymbol{P_d}$ (Eq.4.1, Eq.4.2) (seront détaillées ci-dessous).

L'algorithme proposé par Deneubourg a été repris et étendu par Lumer et Faita [Lumer, 1994] [Khedam,2008] [Ouadfel, 2006] pour la classification des données numériques.

Les travaux de Lumer et Faieta ont inspiré d'autres auteurs pour la résolution de problème de classification par les fourmis. Aussi les travaux de Kuntz et al [Kuntz, 1997] se sont inspirés pour le partitionnement de graphes. Dans [Langham, 1999] un algorithme de classification basé fourmis est proposé pour la minimisation de communication entre les processeurs dans un système de simulation où les traitements sont répartis sur plusieurs processeurs. Dans [Monm, 2000b] Monmarché introduit AntClass un algorithme de classification utilisant des populations de fourmis. AntClass se base sur l'algorithme de Lumer et Faieta avec des modifications de base. AntClass utilise une grille toroïdale et chaque fourmi a la possibilité de transporter plusieurs objets à la fois et de déposer un tas d'objets sur une même case de la grille. De plus AntClass est une hybridation d'un algorithme de fourmis et d'un algorithme de classification classique de type K-means.

AntClust [Ouadfel, 2005] est une autre reprise de AntClass avec des améliorations concernant le support des objets à classer et le déplacement des fourmis était abordée dans la segmentation des images. Pour la seconde approche, l'un des modèles les plus connus (ACO pour Ant Colony Optimization) a été introduit par A. Colorni, M. Dorigo, et V. Maniezzo [Colorni, 1991] initialement dans le cadre du problème du voyageur de commerce. Les fourmis utilisent des phéromones pour marquer des arcs entre les villes. Ces phéromones représentent en fait une distribution de probabilités qui est mise-à-jour en fonction des résultats observés (longueur totale du chemin par exemple).

Pour la troisième approche, la première application ACO appliqué à la classification était celle de Parpinelli et al. (2001, 2002), sont les auteurs qui ont introduit l'algorithme AntMiner pour la découverte des règles de classification [Parpinelli, 2002] [David, 2010]. Plusieurs autres travaux ont été réalisés avec l'utilisation d'ACO pour la classification. Citons C.F. Tsai, C.W. Tsai, H.C. Wu et T. Yang [Tsai, 2004] ont proposé une méthode de partitionnement originale appelée optimisation de colonie de fourmis avec un algorithme de faveur différent (ACO with different favor algorithm ACODF). Dans cet algorithme, une adaptation directe du métaheuristique "ACO" pour résoudre des problèmes de classification [Jafar, 2010]]. P.S. Shelokar, V.K. Jayaraman et B.D. Kulkarni [Shelokar, 2004] ont décrit une méthodologie d'optimisation par colonie de fourmis pour le partitionnement de "N" objets en "K" classes de manière optimale [Jafar, 2010]. S. Le Chi et le C.C. Yang [Chi, 2006] ont développé une nouvelle méthode qui intègre la colonie de fourmis SOM et k-means pour l'analyse de partitionnement. Cet algorithme améliore les résultats des méthodes traditionnelles [Jafar, 2010].

Fei Wang, Dexian Zhang et Na Bao [Monga, 1987] ont proposé une méthode de partitionnement de documents basé sur l'algorithme ACO et l'algorithme de partitionnement Fuzzy C-means (FCM) [Jafar, 2010].

En effet, suivant ce qui a été démontré, les comportements principaux de fourmis utilisés en intelligence artificielles sont :

- Le comportement de tri d'objets et l'auto-assemblage
- La reconnaissance chimique

Ces comportements ont été inspirés par les ingénieurs au problème de classification.

4.5.2. Le comportement de tri d'objets et l'auto-assemblage :

Dans la nature, le**s** fourmis réelles offrent un modèle stimulant pour le problème de classification. La constitution de cimetières, le tri collectif du couvain quand il est dérangé, le rassemblement des œufs en fonction de leur état de développement, et la

recherche du nourriture, sont les exemples les plus marquants.

Quelques travaux expérimentaux montrent que certaines espèces de fourmis sont capables d'organiser spatialement divers éléments du couvain : les œufs, les larves et les nymphes. La figure 4.6 illustre, à travers quatre images, l'expérience menée par Deneubourg et ses collègues [Deneubourg, 1990] qui a consisté à analyser dans un couvain, le comportement des fourmis de l'espèce Messor sancta. L'expérience a duré 26 heures.

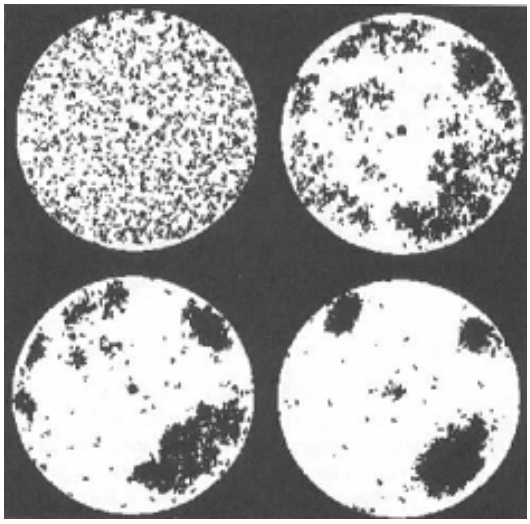


Fig. 4.4 Expérience du tri du couvain chez les Messor sancta. 4 images prises (de gauche à droite) à l'état initial, 2 heures, 6 heures et 26 heures après le début de l'expérience

Ces chercheurs sont arrivés à entrevoir les règles utilisées par les fourmis et qui sont relativement simples : lorsqu'une fourmi rencontre un élément du couvain, la probabilité qu'elle s'en empare est d'autant plus grande que cet élément est isolé. Lorsqu'une fourmi transporte un élément du couvain, elle le dépose avec une probabilité d'autant plus grande que la densité d´éléments du même type dans le voisinage est grande.

4.5.2.1 La reconnaissance chimique

De nombreux algorithmes sont inspirés du comportement de masse des fourmis pour la recherche de nourriture : les phéromones de piste. Chaque fourmi dépose des phéromones à l'aide de son abdomen pour tracer des pistes et suivre un chemin. Il en résulte un phénomène auto-catalytique, en effet, plus une piste est suivie et plus il y a de phéromones et plus il y a de fourmis. Notons que ces phéromones s'évaporent au cours du temps. Les travaux de Deneubourg et al [Deneubourg, 1990] montrent que le plus court chemin pour atteindre une source de nourriture fournira par être emprunté par toutes les fourmis puisque les phéromones sont déposés plus vite sur ce plus court chemin.

La reconnaissance chimique par les phéromones est utilisée dans de nombreuses situations dans lesquelles les fourmis peuvent être confrontées. Par exemple, lorsqu'une fourmi se sent en danger ou a trouvé une proie vivante trop grosse pour elle, elle rejette un nombre important de phéromones, pour que ses congénères puissent remarquer le signal de loin. Les autres fourmis s'approchent donc de la proie, mais le taux de phéromones étant trop fort pour elles, elles restent en retrait. Un nombre important de fourmis va donc s'agglutiner à proximité et lorsque que le taux de phéromones devient

plus faible (dû à l'évaporation), elles attaquent tous ensembles la proie.

Ainsi, la classification fait partie de ces deux problèmes pour lesquels un algorithme à base de fourmis suggère des heuristiques très intéressantes. La première approche se base sur la distribution des données ainsi le traitement sur une grille ou un tableau, tandis que la deuxième se base sur le principe de manipulation des phéromones.

Notre intérêt portera sur le domaine de la classification pour lequel un grand nombre de solutions, dans les différents comportements des fourmis, a été trouvé.

4.5.2.2 Classification à base de fourmis sur une grille

Notations

Les algorithmes qui suivent ont un ensemble de définitions en commun :

- Soit $O = \{ O_1 \ldots O_i \ldots O_N \}$ l'ensemble des objets (i.e. des données, sachant que chaque donnée Contient M attributs) et $|O| = N$ le nombre d'objets.
- Soit une grille G à deux dimensions (G_h x G_l), soit s x s le voisinage d'une case et $r(O_i)$ les objets voisins de l'objet i ;
- Soit n_i le nombre de case dans le voisinage de la case i et contenant un objet.
- Soit $F=\{a_1 \ldots a_A\}$ l'ensemble des fourmis et $|F|=A$ le nombre de fourmis.
- Soit T_{max} le nombre de cycles maximum.
- Soit la probabilité de prendre un objet P_p.
- Soit la probabilité de déposer un objet P_d.
- Soit $f(O_i)$ la mesure de similarité de l'objet i avec ses voisins.

a. Le modèle de Deneubourg [Deneubourg, 1990]

Principe : Ce modèle s'appuie sur la formation de cimetière des Pheidole pallidula. Les objets ont deux attributs qui correspondent leurs coordonnées sur une grille. Chaque fourmi se promène donc aléatoirement sur la grille et ramasse un objet qui lui semble isolé puis le dépose dans une zone dense en objets. Ce modèle est donc une méthode par densité

Soit $f \in [0; 1]$ la densité d'objets perçue par une fourmi (la densité répond à la question « Est-ce un objet isolé ou un tas d'objets ? »), soient k_p et k_d des constantes, alors:

- La probabilité de prendre un objet est **:**

$$P_p = \left(\frac{k_1}{k_1 + f} \right)^2 \qquad \text{Eq.4.1}$$

- La probabilité de prendre un objet est **:**

$$P_d = \left(\frac{f}{k_2 + f} \right)^2 \qquad \text{Eq.4.2}$$

Ainsi, quand $f << k_1$ alors P_1 est proche de 1, et quand $f >> k_1$ alors P_p est proche de 0.

De même,quand $f << k_2$ alors P_d est proche de 0 et quand $f >> k_2$ alors P_d est proche de 1.

Une évaluation possible de f est la suivante : soit *n* le nombre d'objets rencontrées au cours des *T* derniers pas de la fourmi, alors $f = \frac{n}{T}$

Deneubourg et al utilisent les valeurs de $k_1 = 0.1$ et $k_2 = 0.3$.

Cet algorithme de classification par les fourmis a trouvé ses premières applications en robotique collective |Beckers, 1994 ; Martinoli, 1999; Melhuish, 1999].

b. Le modèle Lumer et Faieta [Lumer, 1994]

Principe : Ce modèle est une adaptation du modèle de Deneubourg pour permettre l'exploration de données. L'espace des attributs des données peut ainsi être maintenant supérieure à deux.

La fourmi se promène toujours sur une grille, et peut prendre (respectivement déposer) un objet avec une certaine probabilité déduite de la dissimilarité de l'objet avec ses voisins.

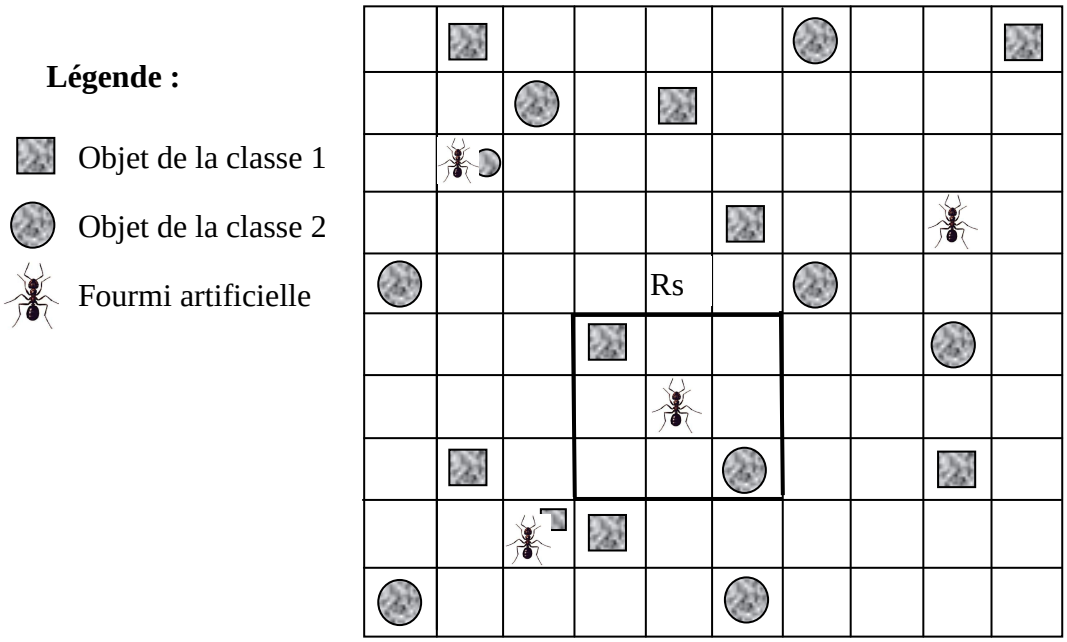


Fig.4.5 Grille de classification de Lumer et Faieta [Labroche, 2003]

- La probabilité de prendre un objet i est identique au modèle de Deneubourg, sauf que c'est la nouvelle mesure de similarité *f(i)* qui est utilisée.

$$P_p(i) = \left(\frac{k_1}{k_1 + f(i)} \right)^2 \qquad \text{Eq.4.3}$$

- La probabilité de déposer un objet i est

$$P_d(i) = \begin{cases} 2f(i) & \text{si } f(i) < k_2 \\ 1 & \text{si } f(i) < k_2 s \end{cases} \qquad \text{Eq.4.4}$$

$$f(i)=\begin{cases}\frac{1}{s^2}\sum_{j\in R_s(r(i))}1-\frac{d(i,j)}{\alpha} & \text{si } f>0\\ 0 & \text{sinon}\end{cases} \qquad \text{Eq.4.5}$$

r(i) est la position de l'objet i sur la grille. f(i) est une mesure de similarité moyenne de l'objet i avec les objets j de son entourage. Le facteur α contrôle la consistance de la fonction de dissimilarité entre les objets. Si α est trop élevé les objets différents seront mis dans la même classe dans le cas contraire les objets similaires ne seront pas regroupés ensembles. Les tests ont été menés avec $k_1=0.1$, $k_2=0.15$, $R=9$, $\alpha=0.5$

L'algorithme 4.1. présente l'algorithme de Lumer et Faita. A fourmis sont utilisées pour effectuer la classification de N objets.

Algorithme 4.1 Algorithme de classification par les fourmis artificielles de Lumer et Faieta

Placer aléatoirement les N objets $o_1,\ldots,o_N$ sur la grille G
Pour T = 1 à T_{max} **faire**
Pour tout aj ∈ $\{a_1,\ldots,a_A\}$ **faire**
Si la fourmi a_j ne transporte pas d'objets et $r(o_i)=r(a_j)$ **alors**
Calculer $f(o_i)$ et $p_p(o_j)$(Eq.4.5)(Eq.4.3)
La fourmi a_j ramasse l'objet o_i suivant la probabilité $p_p(o_i)$ (Eq.4.3)
Sinon
Si la fourmi a_j transporte l'objet o_i et la case $r(a_j)$ est vide **alors**
Calculer $f(o_i)$ et $p_d(o_j)$ (Eq.4.5)(Eq.4.4)
La fourmi a_i dépose l'objet sur la case $r(a_j)$ avec une probabilité $p_d(o_i)$
(Eq.4.4)
Finsi
Finsi
Déplacer la fourmi sur une case voisine non occupée par une autre fourmi a_j
Finpour
Finpour
Retourner l'emplacement des objets sur la grille

Inconvénients et amélioration : Les résultats obtenus ont montré que l'algorithme génère un nombre de classes qui est très souvent très éloigné du nombre réel de classes. Afin de remédier à cela, Lumer et Faieta ont introduit trois extensions au comportement de base des fourmis artificielles :

- Les fourmis se déplacent sur la grille avec une vitesse propre à chacune d'elles comprise entre 1 et $v_{max}=6$. Les fourmis les plus rapides sont moins sensibles aux dissimilarités entre deux objets, ce qui permet de diminuer le nombre de classes générées.

Les fourmis ont différentes vitesses : une vitesse v ∈[1 ; vma_x] est affectée à chaque fourmi suivant une distribution uniforme. Cette vitesse est le nombre de cases parcourues pour un cycle. La mesure de similarité : *f(i)* est transformée :

- Chaque fourmi possède une mémoire à court terme lui permettant de se souvenir des positions des m derniers objets classés. Si une fourmi transporte un objet, elle cherche dans sa mémoire l'objet déjà classé qui est le plus proche de celui qu'elle transporte.

Si elle le trouve, elle se déplacera (avec une certaine probabilité) vers ce nouvel emplacement pour y déposer son objet.

- Si au bout d'un certain nombre d'itérations la fourmi stagne (ne fait plus aucun déplacement) elle peut détruire un groupe en ramassant l'objet le plus éloigné du groupe.

c. Le modèle de Monmarché [Monm, 2000a]

AntClass est un nouvel algorithme de classification non supervisée. Il découvre automatiquement les classes dans des données numériques sans connaître le nombre de classes a priori, sans partition initiale et sans paramétrages délicat. Pour classer des données, ou partitionner un ensemble d'objets, de nombreux algorithmes classiques, tels que les centres mobiles ou Isodata, requièrent qu'une partition initiale soit donnée en entrée. C'est l'inconvénient majeur de ces méthodes : la partition obtenue à partir de cette initialisation risque d'être localement optimale et le seul moyen d'y remédier est de relancer la méthode avec une partition initiale différente. En outre, le nombre de classes exigé par ces méthodes diminue leur intérêt pour un expert cherchant justement à connaître ce nombre de classes.

L'algorithme AntClass utilise les principes exploratoires stochastiques d'une colonie de fourmis. Ces dernières se déplacent sur une grille à deux dimensions et peuvent transporter des objets. La saisie ou le dépôt d'un objet sur un tas dépend de la similarité entre cet objet et les objets du tas [fig.4.6].

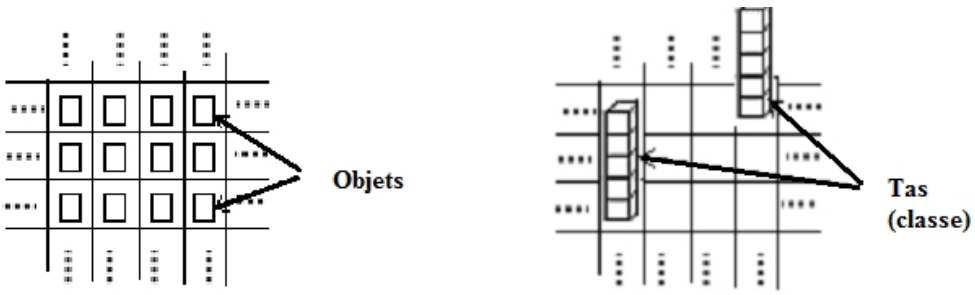


Fig.4.6 Structure d'AntClass. Initialement, chaque cellule de la grille est contituée d'un seul objet. Finalement, après le traitement d'AntClass, les classes sont formées par des Tas. Un Tas est un ensemble d'au moins deux objets et il est localisé sur une case donnée.

L'algorithme AntClass désigne une succession d'itérations des fourmis et des k-moyennes. Les fourmis ont pour tâche de réduire le nombre de classes et les k-moyennes d'améliorer globalement la partition découverte par les fourmis.

Voici la liste des formules nécessaires à l'application de l'algorithme AntClass :

– la distance maximale entre deux objets de l'ensemble O :

$$d^*(O) = \max_{(i,j)\in\{1,\ldots,N\}^2} \{d(\mathbf{x}_i, \mathbf{x}_j)\} \quad \text{Eq. 4.6}$$

– la distance moyenne entre deux objets de l'ensemble O :

$$\bar{d}(O) = \frac{2}{N(N-1)} \sum_{(i,j)\in\{1,\ldots,N\}^2, i<j} d(\mathbf{x}_i, \mathbf{x}_j) \quad \text{Eq.4.7}$$

– la distance maximale entre les objets d'un tas T_j et son centre de gravité $\mathbf{g}_j$:

$$d_g^*(T_j) = \max_{\mathbf{x}_i\in T_j} \{d(\mathbf{x}_i, \mathbf{g}_j)\} \quad \text{Eq.4.8}$$

– la distance moyenne entre les objets d'un tas T_j et son centre de gravité $\mathbf{g}_j$:

$$\bar{d}_g(T_j) = \frac{1}{|T_j|} \sum_{\mathbf{x}_i\in T_j} d(\mathbf{x}_i, \mathbf{g}_j) \quad \text{Eq.4.9}$$

$$p_p(T_j) = \begin{cases} 1 & \text{si } |T_j| = 1 \\ \min\left\{ \left(\frac{\bar{d}_g(T_j)}{d(O)}\right)^{k_1}, 1 \right\} & \text{si } |T_j| = 2 \\ 1 - 0.9\left(\frac{\bar{d}_g(T_j)+\varepsilon}{d_g^*(T_j)+\varepsilon}\right)^{k_1} & \text{sinon} \end{cases} \quad \text{Eq.4.10}$$

$$p_d(o_i, T_j) = \begin{cases} 1 & \text{si } d(\mathbf{x}_i, \mathbf{g}_j) \le d_g^*(T_j) \\ 1 - 0.9\min\left\{ \left(\frac{d(\mathbf{x}_i, \mathbf{g}_j)}{d(O)}\right)^{k_2}, 1 \right\} & \text{sinon} \end{cases} \quad \text{Eq.4.11}$$

L'algorithme 4.2 donne la structure générale de la méthode de classification par les fourmis (appelé Ants par la suite).

L'algorithme 4.3 donne le schéma général de AntClass. L'algorithme K-means est initialisé avec la partition obtenue par Ants.

Le tableau 4.1 résume les paramètres à fixer pour l'algorithme Ants. Ces paramètres sont à spécifier pour chacune des $T_{AntClass}$ itrations de AntClass.

Algorithme 4.2 Algorithme Ants de Monmarché : regroupement des objets par les fourmis. Les indications entre crochets concernent le cas où les fourmis ont une capacité de transport supérieure à 1.

Ants (Grille *G*)

Pour t = 1 à T **faire**
  Pour k = 1 à A **faire**
    Déplacer la fourmi a_k sur une case occupée par une autre fourmi
    Si il y a un tas d'objets T_j sur la même case que a_k **alors**
      Si la fourmi a_k transporte un objet o_i [un tas d'objets T_i] **alors**
        Déposer l'objet o_i [le tas T_i] transporté par la fourmi sur le tas T_j suivant la probabilité $p_d(o_i,T_j)$[$p_d(T_i,T_j)$](Eq.4.11)
      Sinon
        /* La fourmi ne transporte pas d'objet */
        Ramasser l'objet oi le plus dissimilaire du tas T_j [jusqu'à ce que la capacité $c(a_k)$ de la fourmi soit atteinte ou que le tas soit vide] selon la probabilité $p_p(T_j)$ (équation 4.10)
      FinSi
    FinSi
FinPour
FinPour Retourner la grille G

Et voici l'algorithme de Monmarché AntClass:

Algorithme 4.3 Algorithme *AntClass* de classification non supervisée par des fourmis et les k-moyennes de Monmarché.

AntClass ()

Soit P_0 la partition initiale formée de N classes.
  Pour t = 1 à $T_{AntClass}$ **faire**
    Initialiser la grille G à partir de la partition P_{t-1} (un tas par classe)
    G' ←Ants(G)
    Construire la partition P' associée à la grille G'
    P_t ←K-Means(P')
  FinPour
Retourner la partition $P_{T_{AntClass}}$

Tableau 4.1 Paramètres de l'algorithme Ants

Paramètre	Description
A	Nombre de fourmis
T	Nombre de déplacements de chaque fourmi
$c(a_i)\ \forall i \in\{1,\ldots,A\}$	Capacité de transport des fourmis
$m(a_i)\ \forall i \in\{1,\ldots,A\}$	Taille de la mémoire de chaque fourmi
$v(a_i)\ \forall i \in\{1,\ldots,A\}$	Vitesse sur la grille de chaque fourmi
$p(a_i)\ \forall i \in\{1,\ldots,A\}$	Patience de chaque fourmi
k_1,k_2	Paramètre de calcul des probabilités p_p et p_d

Remarque : Les deux derniers modèles, de Lumer et Faieta et de Monmarché, présentent leurs classes sous forme d'un ensemble d'objets de différentes formes. Pour le premier est un amas d'objet et pour le second est un Tas d'objets [Fig.4.7].

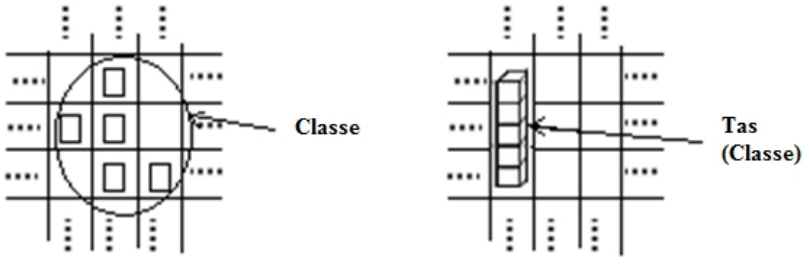


Fig.4.7 Représentation des classes. Sur la gauche, nous montrons une classe selon Lumer et faieta (1994). A droite, selon Monmarché.

Cette figure montre qu'une classe trouvé selon Lumer et Faieta a alors une certaine superficie et deux classes peuvent être en contact et posent un problème pour les séparer. Mais suivant monmarché ce problème est totalement résolu avec la faciliter de définir une classe.

4.5.2.3 Classification à base de fourmis sur un tableau : AntClust [Ouadfel, 2006]

Le modèle a repris les travaux de Lumer et Faieta et ceux de Monmarché pour en améliorer les points concernant le support des objets à classer et les déplacements des fourmis.

Dans la plupart des algorithmes de classification basés fourmis, les objets sont placés sur une grille à deux dimensions et les fourmis se déplacent sur la grille d'une case à une autre et utilisent une mesure de similarité locale pour regrouper des objets de même nature. Dans AntClust [Ouadfel, 2006], la grille est abandonnée car plusieurs paramètres s'y rattachent et il n'est pas facile de trouver le paramétrage adéquat. Nous citons en particulier :

- La taille de la grille qui a une grande influence sur la convergence de l'algorithme. La grille ne doit pas être trop grande car les fourmis vont perdre du

temps à chercher les objets, ni trop petite sinon il n y aura pas de cases vides pour déposer les objets déplacés par les fourmis.

- Chaque case de la grille ne peut contenir qu'un seul objet à la fois. Ce qui signifie qu'une fourmi peut passer un certain temps à trouver une case libre sur la grille.

- Le déplacement des fourmis sur la grille étant aléatoire, certaines cases peuvent ne pas être visitées par les fourmis et donc les objets qui y sont placés ne seront pas ramassés en un nombre d'itérations acceptables.

- Les résultats obtenus sont essentiellement visuels. Il faut passer par un post traitement pour les exploiter en une partition d'objets.

Pour ces différentes raisons, la grille est abandonnée car plusieurs paramètres s'y rattachent et il n'est pas facile de trouver le paramétrage adéquat.

Dans AntClust, l'environnement des fourmis est un tableau de N cellules [Fig.3.8] reliées chacune à un emplacement représentant le nid de la colonie, afin de faciliter les déplacements des fourmis d'une cellule à une autre [Fig.3.9].

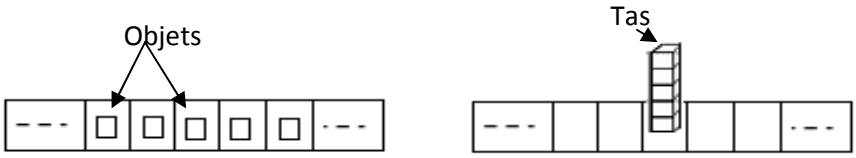


Fig.4.8 Structure de données de AntClust.

Initialement, chaque cellule est constituée avec un unique objet. Finalement, dans AntClust les classes sont formées avec des Tas.

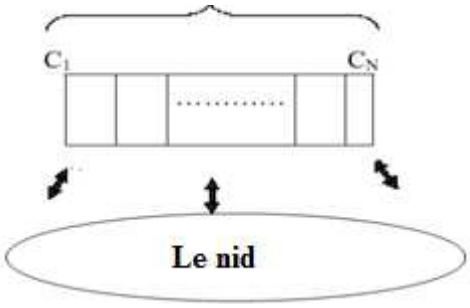


Fig.4.9 L'environnement des fourmis artificielles

Initialement les N pixels de l'image I à classifier sont placés sur le tableau de telle sorte qu'une cellule ne contienne qu'un pixel à la fois. Durant le processus de classification une cellule peut correspondre à un ou plusieurs pixels de l'image d'origine. A la fin de l'algorithme le nombre de cellules non vides représente le nombre possible de classes présentes dans l'image.

On considère un ensemble de N pixels $\{p_1, p_2,\ldots,p_N\}$que l'on désire regrouper en des classes aussi homogènes que possible en terme de niveau de gris. Nous considérons aussi une population A de K fourmis $\{a_1,a_2,\ldots,a_K\}$ qui coopèrent ensemble et communiquent par stigmergie pour fournir une classification optimale.

Initialement on a N classes constituées chacune d'un pixel. Au cours du processus de classification les fourmis se déplacent les pixels d'une classe à une autre

et tentent de regrouper dans une même classe le maximum de pixels similaires en terme de niveau de gris.

Pour cela, nous devons évaluer une mesure de similarité entre un pixel p_i de niveau de gris ng_i et le centre de gravité g_k d'une classe c_k définie comme suit:

$$f(p_i, c_k) = \frac{1}{1 + \left(\frac{ng_i - g_k}{\beta} \right)^2} \qquad \text{Eq.4.12}$$

β est un paramètre qui contrôle la dilatation de la fonction *f*.

Un schéma représentant l'évolution de la fonction *f* est donné dans la figure (Fig.3.10) [Ouadfel, 2006].

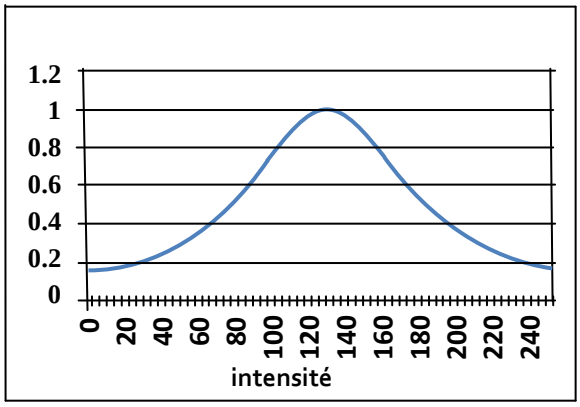


Fig.4.10 Schéma de la fonction de similarité pour ng_i =128 et g_k ε [0,255], β =50

La fonction de similarité f(.) atteint son maximum pour $ng_i = g_k$ et est normalisée entre 0 et1.

Durant le processus de classification les fourmis se déplacent périodiquement de leur nid vers un tableau de *N* cases représentant des classes de pixels. Ce tableau possède les propriétés suivantes :

- Chaque case du tableau est reliée au nid des fourmis ce qui facilite le déplacement des fourmis sur le tableau,
- Chaque case du tableau peut contenir un nombre illimité de pixels similaires en terme d'une mesure de similarité,
- Initialement il y'a sur le tableau autant de case que de pixels à regrouper et chaque case ne contient qu'un seul pixel.

Par rapport à la grille utilisée dans les précédents travaux, ce tableau permet deux principaux avantages:

- Il permet de s'assurer que les fourmis ne vont pas perdre de temps à chercher les pixels sur la grille ;
- L'identification des classes de pixels est immédiate du fait qu'une case peut contenir plus qu'un pixel, alors que dans les autres travaux, une classe est représentée par un amas d'objets qui peuvent se toucher ce qui rend l'extraction des classes difficiles et ambigües.

Dans ce qui suit, le terme « case » désignera une classe de pixels

L'algorithme AntClust

L'algorithme commence avec une phase initiale dans laquelle (1) les N pixels sont placés aléatoirement sur les cases du tableau; (2) les A fourmis $\{a_1,a_2,\ldots,a_K\}$sont déplacées de leur nid et disposées aléatoirement sur les cases du tableau en vérifiant qu'une case ne peut contenir qu'une seule fourmi à la fois ; et (3) la fourmi ramasse un pixel de la case où elle se trouve. A la suite de cette étape, le processus de classification commence : c'est une boucle simple, dans laquelle (1) une fourmi est sélectionnée aléatoirement ; (2) elle revient vers son nid et se déplace vers une case guidée par une information heuristique; et (3) la fourmi décide d'y déposer le pixel qu'elle transporte selon une règle probabiliste. Une fois qu'elle devient libre, elle effectue de nouveaux déplacements entre le nid et les cases du tableau afin de rechercher le prochain pixel à transporter. Le ramassage d'un pixel est aussi effectué selon une règle probabiliste. Cette boucle est répétée pour chacune des fourmis.

Au cours du processus de classification, aucune nouvelle classe n'est crée mais une classe peut disparaître si la case qui correspond du tableau se vide de ses pixels. A la fin du processus de partitionnement, le nombre de classes intéressantes de l'image correspond au nombre de cases non vides présentes sur le tableau.

Algorithme 4.5 L'algorithme de classification AntClust stochastique[Ouadfel, 2006]

AntClust()

```
/* Initialisation*/
Pour chaque pixel p_i faire
    Placer p_i dans une cellule du tableau
Finpour
Pour chaque fourmi a_1 faire
     Placer a_1 dans une cellule choisie aléatoirement et lui affecter son pixel;
     Etat [a_1] := porteuse ;
Finpour
Déplacer toutes les fourmis vers le nid
/* Boucle principale*/
Pour t=1 à t_max faire
  Pour chaque fourmi a1 faire
    Si état[a_1]=porteuse alors
      Déplacer a_1 vers une cellule ck
      Dépôt := faux ;
      Dépôt := déposer le pixel p_i qu'elle transporte dans c_k avec une
                probabilité
     P_d(p_i,c_k) (Eq.4.14)
     Si Dépôt = vrai alors
       Etat [a_1] : = libre ;
     Finsi
     Sinon
       Choisir aléatoirement un pixel p_i ;
       Déplacer a_1 vers la cellule c_k contenant p_i ;
       Porter := faux ;
       Porter : = Porter p_i de sa cellule avec une probabilité P_p (p_i,c_k),
                                                              (Eq.4.13)
```

```
   Si Porter = vrai alors
                  Etat [a1] := porteuse ;
               Finsi
            Finsi
            Déplacer les fourmis vers le nid
         FinPour
       Finpour
Retourner la partition obtenue
```

Dans la suite de ce paragraphe, nous allons décrire en détail les règles de déplacements, de ramassage et de dépôt de pixels que les fourmis vont utiliser sur le tableau pour classer les pixels de l'image.

a. Déplacements des fourmis

Durant le processus de partitionnement, les fourmis vont se déplacer régulièrement entre leur nid et les cases du tableau pour transporter ou bien déposer un pixel. Afin d'accélérer le processus de regroupement et donc la convergence de l'algorithme, la fourmi n'a pas un mouvement complètement désordonné. Pour cela une version modifiée du mécanisme de mémoire à court terme introduit dans [Lumer, 1994] et [Monm, 1999] est proposée.

Dans l'approche de Lumer et Faieta, chaque fourmi mémorise les m derniers objets qu'elle a ramassé ainsi que leurs emplacements sur la grille. A chaque fois qu'elle ramasse un nouveau objet, il est comparé aux objets contenus dans sa mémoire. Elle se dirige après vers l'emplacement de l'objet qui lui est le plus similaire en terme de distance euclidienne. Ce mécanisme a été étendu dans [Handl, 2003] en remplaçant la distance euclidienne entre deux objets par la fonction de voisinage appliquée aux positions de tous les objets à classer. Monmarché reprend les idées de Lumer et Faieta et utilise la distance entre le centre de gravité du tas transporté par la fourmi et les tas qu'elle a mémorisé (puisque dans son approche, les fourmis peuvent transporter plus qu'un objet à la fois) pour choisir le prochain emplacement de l'objet (ou du tas) qu'elle transporte. Nous adaptons ces idées pour la classification des images et l'étendons pour le dépôt et le ramassage de pixels comme suit:

Quand une fourmi transporte un pixel, on l'autorise à accéder à son voisinage immédiat. Elle calcule la fonction de similarité définie dans Eq.4.12 pour chacune des cases des 8 voisins du pixel qu'elle transporte et évalue ainsi directement la possibilité de le déposer dans une des ses cases candidates. Le meilleur emplacement sera celui pour lequel la fonction de similarité est maximum. La fourmi décide alors de déposer son pixel sur cet emplacement avec une probabilité $\boldsymbol{p_d}$. Si cette décision est négative, la fourmi garde le pixel qu'elle transporte, et essaye d'autres cases choisies aléatoirement jusqu'à ce qu'elle arrive à le déposer.

Quand une fourmi recherche un pixel à transporter, cette recherche est faite en utilisant un index commun contenant les pixels libres (non transportés par une fourmi). Nous avons choisi de trier l'index par ordre croissant en fonction de la distance entre le niveau de gris du pixel et le centre de gravité de la classe où il se trouve. Ce choix présente deux avantages : (1) il assure de ne transporter que les pixels les plus éloignés des centres de gravités des classes à laquelle ils appartiennent (les pixels les plus dissimilaires) et (2) facilite la mise à jour de l'index à chaque fois qu'une fourmi dépose

un pixel dans une case du tableau. Initialement, l'index contient tous les pixels de l'image.

b. Ramassage d'un pixel

La probabilité de transporter un pixel p_i de sa case c_k est définie par la formule suivante :

$$p_p(p_i, c_k) = \begin{cases} 1 & \text{si } |c_k| = 1 \\ q & \text{si } |c_k| = 2 \\ \dfrac{k_p}{k_p + f(p_i, g_k)} & \text{sinon} \end{cases} \qquad \text{Eq.4.13}$$

où $| c_k |$ est le nombre de pixels dans la case c_k et g_k son centre de gravité. Si la classe c_k ne contient qu'un seul pixel, il est systématiquement ramassé par la fourmi. Si la classe contient deux pixels, la fourmi a une probabilité q de ramasser le pixel p_i. Enfin si la case contient plus de deux pixels la probabilité p_{porter} de transporter le pixel p_i est importante quand la fonction de similarité entre le centre de la classe c_k et le niveau de gris du pixel p_i est faible (tend vers 0).

c. Dépôt du pixel

Si une fourmi transporte un pixel p_i, elle explore son voisinage immédiat pour choisir la case c_k vers laquelle elle se déplacera pour y déposer avec une probabilité donnée par la formule suivante :

$$\begin{cases} 1 & \text{si } f(p_i, g_k) \le f(p_{dissim}, g_k) \\ p_d(p_i, c_k) = \dfrac{f(p_i, g_k)}{f(p_i, g_k) + k_d} & \text{sinon} \end{cases} \qquad \text{Eq.4.14}$$

Avec :

$$f(p_{dissim}, g_k) = \min_{p_i \in k} (f(p_i, g_k)) \qquad \text{Eq.4.15}$$

Ainsi si le pixel transporté par la fourmi est plus proche du centre de la classe c_k que le pixel le plus éloigné de cette classe il est y déposé. Sinon, plus la fonction de similarité entre p_i et c_k est petite, moins la probabilité de dépôt sera faible.

d. Mémoire des fourmis

Durant le processus de classification, les fourmis vont se déplacer régulièrement entre leur nid et les cellules du tableau pour transporter ou bien déposer un pixel. Afin d'accélérer le processus de regroupement et donc la convergence de l'algorithme, la fourmi n'a pas un mouvement complètement désordonné. Pour cela une version modifiée du mécanisme de mémoire à court terme introduit dans [Lumer 1994] et [Monm 1999] est proposée. Dans l'approche de Lumer et Faieta, chaque fourmi mémorise les m derniers objets qu'elle a ramassés ainsi que leurs emplacements sur la grille. A chaque fois qu'elle ramasse un nouvel objet, il est comparé aux objets contenus

dans sa mémoire afin de biaiser la direction qu'elle va prendre. La fourmi a tendance à se diriger vers l'emplacement où elle a déposé auparavant un objet similaire à celui qu'elle transporte actuellement). Ce mécanisme a été étendu dans [Handl, 2003] en remplaçant la distance euclidienne entre deux objets par la fonction de voisinage appliquée aux positions actuelles de tous les objets contenus dans la mémoire de la fourmi. Monmarché reprend les idées de Lumer et Faieta et remplace la comparaison des objets sur la distance les séparant par la distance entre le centre de gravité du tas transporté par la fourmi et les tas qu'elle a mémorisé (puisque dans son approche, les fourmis peuvent transporter plus qu'un objet à la fois) pour choisir le prochain emplacement de l'objet (ou du tas) qu'elle transporte. Dans les précédents travaux, la mémoire de la fourmi est statique, ce qui implique qu'elle ne reflète pas les changements des positions des objets qu'elle a transporté. Dans ce cas, il se peut que la fourmi revienne vers un objet qui a changé de position. Nous adaptons ces idées pour la classification des images et l'étendons pour le dépôt de pixels comme suit :

Chaque fourmi mémorise les *m* derniers objets qu'elle a ramassés ainsi que leurs emplacements sur tableau. Quand une fourmi transporte un pixel, elle consulte sa mémoire et évalue la possibilité de le placer dans la cellule d'un des pixels qu'elle a déjà transporté. Pour cela, elle calcule la fonction de similarité *f( )* pour chacune des cellules mémorisées. La cellule candidate à recevoir le pixel p_i sera celle pour laquelle la fonction *f* est maximum. La fourmi se dirige de son nid avec la cellule candidate et décide d'y déposer son pixel avec la probabilité p_d . Si cette décision est négative, sa mémoire est désactivée et dans les prochaines itérations, elle tentera de déposer son pixel dans une autre cellule choisie aléatoirement, jusqu'à ce qu'elle y arrive.

4.5.2.4 Classification à base de population phéromones de fourmis

a. Introduction

L'optimisation par colonies de fourmis est inspirée par la façon dont les fourmis cherchent leur nourriture.

L'un des modèles les plus connus (ACO pour Ant Colony Optimization) a été introduit par [Colorni, 1991] initialement dans le cadre du problème du voyageur de commerce. Les fourmis utilisent des phéromones pour marquer des arcs entre les villes. Ces phéromones représentent en fait une distribution de probabilités qui est mise à jour en fonction des résultats observées (longueur totale du chemin par exemple). Cette approche a été depuis largement développée et appliquée à de nombreux problèmes d'optimisation combinatoire et numérique. La classification est un problème d'optimisation dont il y a souvent des optima locaux des critères à optimiser. C'est pour cela que l'implémentation des heuristiques modernes d'optimisation combinatoire telle que l'algorithme d'optimisation de colonies de fourmis peut être intéressante (ACO : Ant Colony Optimization).

Cette heuristique a montré des bonnes performances dans plusieurs problèmes d'optimisation (voir chapitre 3 pour plus de détaille).

La classification à base de phéromone était en premier lieu proposée par Parpinelli et al. (2001, 2002) qui ont ouvert la porte à ce domaine. Parpinelli et al. ont introduit l'algorithme AntMiner pour la découverte des règles de classification [Parpinelli, 2002] [David, 2010]. Ne pas oublier par suite la proposition faite par Monmarche dans : « Compétition de colonies de fourmis pour l'apprentissage supervisée : CompetAnts» en 2002.

L'algorithme AntMiner est une règle d'induction ou de classification utilisant la méta-heuristique « Ant Colony Optimization ». L'algorithme AntMiner a été observé prématurément convergent sur les solutions à certains situations. L'algorithme a également été observée à avoir des difficultés de convergence sur les règles.

Seulement que ces deux algorithmes, AntMiner et CompetAnts sont adaptés simplement au principe des phéromones pour résoudre un problème d'apprentissage supervisée, c'est-à-dire pour lequel un étiquetage des données est connu ce qui est hors de l'objectif de notre étude.

Lors de notre recherche bibliographique, nous n'avons trouvé aucun article qui traite de la méthode des colonies de fourmis orienté phéromone pour la classification des images. Seulement qui ont proposé des méthodes de classification pour d'autres types de données [Yucheng, 2006] [Handl, 2006] [Trejos, 2004] [Shelokar, 2004] [Jafar, 2010] [Tsai, 2004] [Kao,2006] [Farhad,2010].

Dans [Trejos, 2004], l'approche utilisée pour classifier des données issues de la base de données « Machine learning Repository UCI » [MLRUCI, 2012] est l'algorithme qui est nommé AcoClus.

Cependant, les résultats obtenus par l'algorithme AcoClus ont montrés qu'elles sont supérieurs à ceux de plusieurs autres heuristiques.

Notre travail s'est basé sur les travaux de [Trejos, 2004] qui a pu amélioré le travail de [Parpinelli, 2002] et l'a utilisé pour une classification non supervisée.

b. Partitionnement par colonies de fourmis AcoClus

Cette partie se base sur ce qui a été vu dans le chapitre 3 concernant la métaheuristique *ACO*.

Malgré ça, on va reprendre quelques notions pour permettre de la lecture aisée des étapes qui vont suivre.

b.1 La méta-heuristique *ACO*

Les **algorithmes de colonies de fourmis** sont des algorithmes inspirés du comportement des fourmis et qui constituent une famille de méta-heuristiques d'optimisation. L'algorithme d'Optimisation par Colonies de Fourmis (Ant Colony Optimiztion : *ACO*) est un cas particulier de ces algorithmes. Initialement proposé par Marco Dorigo *et al.* dans les années 1990 [Colorni, 1991] [Dorigo, 1992], pour la recherche de chemins optimaux dans un graphe. Il estime que rechercher une source de nourriture est analogue à rechercher une solution dans un espace de recherche commun.

L'algorithme général d'*ACO*, dans le cas où les états sont noté par *i*,*j*, sur un graphe de *n* nœuds, est le suivant [Trejos, 2004] [Trejos, 2007] :

Algorithme 4.6 L'algorithme *ACO* de Trejos:

Initialiser $\tau_{ij} = \tau_0$;
Dépot de chaque fourmi dans un noeud
Pour t = 1 jusqu'à t_{max} **faire**
  Pour m = 1 jusqu'à M, **faire**
    Construire une solution $S^m(t)$
    Calculer le cout $\zeta^m_{intra}(P)$ de $S^m(t)$
  Finpour

```
        Pour chaque arc (i,j) faire
            Mise-à-jour τ
        Finpour
Finpour
```

L'*ACO* a été conçue pour résoudre les problèmes d'optimisation discrets. Bien que la méta-heuristique soit relativement récente, son applicabilité a été élargie pour résoudre divers problèmes combinatoires statiques. Parmi ceux recensés dans la littérature, nous citons: *coloriage des graphes*, *assignement quadratique*, *ordonnancement des tâches* [Jodogn, 2001] ainsi le problème de partitionnement.

Pour sélectionner le nœud suivant du chemin, la probabilité de l'état de transition est définie comme suit:

$$p_{ij}(t) = \frac{\left[\tau_{ij}(t)\right]^{\alpha}\left[\eta_{ij}(t)\right]^{\beta}}{\sum_{l=1}^{n}\left[\tau_{ij}(t)\right]^{\alpha}\left[\eta_{ij}(t)\right]^{\beta}} \qquad \text{Eq.4.16}$$

Voir Eq.3.3

Où τ_{ij} et $1/\eta_{ij}$ sont l'intensité de phéromone et la taille du chemin entre les nœuds *i* et *j*, respectivement.

α et β sont des paramètres de contrôle et apportent un équilibre entre la visibilité et l'intensité.

A l'instant *t*, les fourmis font un compromis entre la visibilité et l'intensité de la trace des phéromones présentes sur tous les chemins pour choisir la prochaine destination *j*.

Afin de satisfaire la contrainte "*parcourir touts les objets en empruntant le plus court chemin*", on associe à chaque fourmi une structure appelé *liste taboue*. Cette liste va contenir les libellés des objets visités auparavant et permet d'éviter à la fourmi d'atteindre une destination déjà rencontrée durant le cycle. A la fin du cycle cette liste est remise à zéro.

Après le choix du prochain chemin, l'intensité de la trace de phéromone est mise-à-jour comme suit :

$$\tau_{ij}(t+1) = \rho\tau_{ij}(t) + \Delta\tau_{ij} \qquad \text{Eq.4.17}$$

tel que:

$$\Delta\tau_{ij}^{k} = \begin{cases} 1/L_k & \text{si la fourmi parcours un nœud (i, j)} \\ 0 & \textit{Sinon} \end{cases} \qquad \text{Eq.4.18}$$

Où $\boldsymbol{L_k}$ est la longueur de cycle associé à la $k^{ème}$ fourmi.

On peut trouver les détails de cette heuristique dans [Bonab, 1999].

b.2 Description d'AcoClus

L'optimisation par colonies de fourmis est inspirée par la façon dont les fourmis cherchent leur nourriture. ACOClus est un algorithme hybride composé des deux algorithmes k-means et ACO itératif tel qu'à chaque itération on examine toutes les fourmis.

Les phases de l'algorithme ACOClus :

- **Initialisation des paramètres** : plusieurs paramètres doivent êtres initialisés qui sont :
  - Le taux de phéromone entre tout les objets *i* et *j* est initiale à τ_0;ce taux est choisi d'une façon empirique.
  - Calcul de la visibilité entre tous les objets *i* et *j* : η
  - Initialiser au hasard les partitions $P^1, .. , P^M$ associées à chaque fourmi
- **Application de k-means :**

Au début, une fourmi *m* est associée à une partition P^m générée au hasard, on applique k-means et on converge à un minimum local de $\zeta_{intra}(P)$.

Ce processus se répète pour la totalité des *M* fourmis.

- **Application d'ACO :**

Pendant les itérations, la fourmi m modifiera la partition P^m comme suit: un objet *i* est choisi au hasard, et un autre objet *j* est sélectionné au hasard selon une stratégie avec probabilité p_{ij}, où p_{ij} dépend de la trace de phéromone et de l'heuristique locale. On peut dire que la fourmi décide si *j* sera affecté à la même classe que *i*.

Si on note *t* l'itération en cours, la partition associée à la fourmi *m* sera notée $P^m(t)$. La valeur de la trace de phéromone est modifiée selon la règle *:*

$$\tau_{ij}(t+1) = (1-\rho)\,\tau_{ij}(t) + \rho \sum_{m=1}^{M} \Delta^m \tau_{ij}(t+1) \qquad \text{Eq.4.19}$$

Où τ_{ij} associe deux objets *i, j* de Ω, et $\rho \in$]0, 1] est un *coefficient d'évaporation*.

$\Delta^m \tau_{ij}(t+1)$ est la quantité de phéromone par fourmi dans l'association des objets *i et j* dans la même classe, définie par :

$$\Delta^m \tau_{ij}(t+1) = \begin{cases} \dfrac{\zeta_{inter}(P^m(t))}{I} & \text{si i et j appartiennent à la même classe de } P^m \\ 0 & \textit{Sinon} \end{cases} \qquad \text{Eq.4.20}$$

I est l'inertie totale : $I(P) = \zeta_{intra}(P) + \zeta_{inter}(P)$

Si on considère que les objets manipulés sont des points qui forment un nuage alors *I* est la somme des distances de tous les points au centre de gravité globale de ce nuage.

Donc si les objets i et j sont dans une même classe d'une partition P^m on aura :

$$\Delta^m \tau_{ij}(t+1) = \frac{\zeta_{inter}(P^m(t))}{\zeta_{inter}(P^m(t)) + \zeta_{intra}(P^m(t))} = 1 + \frac{\zeta_{inter}(P^m(t))}{\zeta_{intra}(P^m(t))}$$

Et dans le cas où les objets n'appartiennent pas aux mêmes classes, cette quantité de phéromone dans cette association est nulle.

$$\eta_{ij} = \frac{1}{d(x_i, y_j)} \qquad \text{Eq.4.21}$$

$\zeta_{inter}(P^m(t))$ étant l'inertie interclasses de la partition $P^m(t)$. Deux objets classifiés dans la même classe laissent donc la trace de phéromone. L'heuristique locale ou visibilité à court terme est définie par la formule (Eq. 4.21) de façon à ce que deux objets proches ont une influence de la probabilité de les affecter dans la même classe. Soit α et β deux paramètres réels positifs. Si une fourmi *m* est placée sur un objet *i*, l'objet *j* est choisi avec la probabilité (Eq.4.16) et on introduit *j* dans la même classe que *i*.

La matrice de probabilité $(p_{ij}(t))_{n\times n}$ est définie avant l'opération de déplacement des fourmis entre les objets et possibilité de leur affectation à une classe.

$$p_{ij}(t) = \begin{bmatrix} 0.1 & 0 & ... & 0.2 & 0 \\ 0 & 0.2 & ... & 0.1 & 0.1 \\ ... & ... & ... & ... & ... \\ 0.2 & 0.2 & ... & 0.2 & 0 \\ 0.2 & 0.1 & ... & 0.2 & 0.1 \end{bmatrix}_{nxn} \qquad \text{Eq.4.22}$$

Les lignes de la matrice $(p_{ij}(t))_{n\times n}$ somment 1; étant donné *i*, la valeur $p_{ij}(t)$ est la probabilité de choisir *j*, qui est modelée en utilisant la probabilité cumulée et en générant des nombres pseudo-aléatoires. Cette probabilité sera meilleure que *j* est affecté à la classe de *i*.

En considérant les éléments ci-dessus, l'algorithme AcoClus est le suivant:

Algorithme 4.7 L'algorithme AcoClus de Trejos **[Trejos, 2007]**

AcoClus()

Initialiser $\tau_{ij} = \tau_0$;
Calculer η (Eq.4.21)
Initialiser les probabilités *p*
Initialiser au hasard les partitions $P^1, .. , P^M$ associées à chaque fourmi
Par k-means sur chaque P^m converger vers un minimum local de ζ_{intra}
Pour *t* = 1 jusqu'à t_{max} faire
  Pour *m* = 1 jusqu'à *M*, faire
    Choisir au hasard un objet *i*
    Choisir un objet *j* avec probabilité $p_{ij}(t)$
    Affecter *j* à la classe de *i*
  FinPour
  Calculer $\zeta_{inter}(P^1), ..., \zeta_{inter}(P^M)$ et garder la meilleure valeur
  Mettre à jour $\tau(t)$
FinPour

Finalement, l'algorithme AcoClus a les six (06) paramètres suivants:

1- le nombre *M* de fourmis,
2- la valeur initiale de phéromone τ_0,
3- le nombre maximal d'itérations t_{max},
4- le poids α de la phéromone dans Eq.4.16,
5- le poids β de l'heuristique locale en Eq.4.16
6- et le coefficient ρ de l'évaporation dans Eq.4.19.

Remarque:
La sélection de l'objet *j*: Pour déterminer *j* pour un *i* sélectionné, il existe plusieurs stratégies qui peuvent être appliquées:

a. Roulette aléatoire: Ce choix aléatoire est semblable à ce qu'on appelle la roulette aléatoire dans les algorithmes génétiques **[Trejos, 2004] [Trejos, 2007]**.
b. Permettre aux fourmis de se déplacer d'une manière gourmande à un nœud dont le produit du niveau de phéromone et la valeur heuristique est le plus élevé (voir l'équation (Eq.4.23)), avec une probabilité définie à priori q_0 et une probabilité générée aléatoirement q **[Kao,2006]**.
c. Attribution des probabilités aux nœuds des candidats, puis permettre à la fourmi choisir l'un d'entre eux d'une manière stochastique selon l'équation (Eq.4.16). Le nœud le plus attrayant est à cause de sa plus grande probabilité dont il dispose **[Kao,2006]**.

$$j = \begin{cases} \arg \max\left\{\left[\tau_{ij}(t)\right]^{\alpha}\left[\eta_{ij}(t)\right]^{\beta}\right\} si\ q \le q_0 \\ S \qquad\qquad Sinon \end{cases} \qquad \text{Eq.4.23}$$

Pour notre cas, on a opté pour la dernière possibilité.

b.3 Complexité de l'algorithme ACOClus :

Afin de se faire une idée, nous divisons l'algorithme **Algorithme 4.7,** ci-dessus, pour calculer la complexité:

1. Initialisation.
2. Application de K-means pour chaque partition
3. Déplacement de chaque fourmi dans une partition
4. Calcul des dépôts de phéromone.
5. Calcul des inerties inter-classes pour chaque partition
6. Boucle de l'algorithme.

Procédons étape par étape :

1. Initialisation de l'algorithme. Les éléments s'agencent de la manière suivante au début de l'algorithme :

   1.1 Les *m* fourmis sont répartis aléatoirement sur les *n* objets.
   1.2. Les pistes de phéromones sont initialisées comme suit : $\tau_{ij} = \tau_0$, où τ_0 est une petite constante positive, qui ne peut être nulle (sinon, il y a un problème lors du calcul de (Eq.4.16))

1.3 Calcul de la visibilité η afin de procéder aux différents prochains calculs de la probabilité p_{ij}. Pour cela, on doit préparer une matrice de dimension N pour calculer la distance entre tous les pixels de l'image.

$$\eta = \begin{vmatrix} 1/d_{1,1} & \dots & 1/d_{1,n} \\ \dots & \dots & \dots \\ 1/d_{n,1} & \dots & 1/d_{n,n} \end{vmatrix} \text{ on se réfère à la formule du } \eta \qquad \text{Eq.4.24}$$

Cette matrice est carrée et symétrique.
Ce calcul global de la matrice des distances s'effectue à chaque fois lorsqu'on passe aux calculs des probabilités. Seulement, il est possible de l'éviter surtout quand on opte à choisir les objets j à partir d'un objet i et on calcul leurs probabilités p_{ij}. Donc, on calcul seulement une seule ligne.

1.4 *Initialiser les probabilités p :* On passe directement au calcul des probabilités et ceci après calcul des visibilités η.

1.5 Initialiser au hasard les partitions $P^1, .. , P^M$ associées à chaque fourmi

2. Application de l'algorithme *k-means* sur chaque partition P^m qui converge vers un minimum local de ζ_{intra}

3. Fin d'un cycle. Après toute itération, chaque fourmi déplace un objet à une classe suivant une probabilité. À ce moment les variables $\tau_{ij}^{k}(t)$ et les intensités de phéromone au cycle t associé à la $k^{ème}$ fourmi sont calculées conformément à la formule (Eq.4.16).
4. Les variables de phéromone $\tau_{ij}(t)$ sont mises à jour suivant la formule (Eq.4.19).
5. *Calcul des inerties inter classe* $\zeta_{inter}(P^1),\dots,\zeta_{inter}(P^M)$ *et sauvegarde de la meilleure valeure.*

Donc, le calcul global de la complexité est le suivant :

1. On a une complexité $O(M)+O(N^2)+O(N)$: Initialisation des M fourmis, calcul de la matrice des distances et initialisation des probabilités.
2. La complexité de k-means $O(NKT)$. Du moment on a M partitions, donc on a M k-means *: $O(MNKT)$*
3. La complexité est $O(M).O(N)$, puisque chaque fourmi se déplace à une classe après calcul des probabilités d'une seule ligne de la matrice p_{ij} (O(N)).
4. La complexité est $O(TM)$: pour chaque itération toutes les fourmis sont traitées.
5. La complexité $O(K)$: K qui est le nombre de classes qu'on calcule les inerties inter-classes.

Soit la complexité : $O(M) + O(N^2) + O(N) + O(MNKT) + O(T).O(M).O(N^2) + O(TM) + O(T).O(K)$ qui est approximé à $O(N^2)+O(MNKT)+O(TMN)$

La complexité ACOClus au meilleur des cas devienne ***O(MNKT).***

On remarque que cette complexité est fonction de la complexité de k-means suivant le nombre de fourmis ***O[M(NKT)]***.

L'algorithme ACOClus présente les caractéristiques suivantes :

- Chaque fourmi est associée à une partition,
- A chaque traitement d'une partition on calcul les inerties.

A travers ces points qui sont couteux en termes de temps de calcul et par suite en complexité algorithmique, l'algorithme ACOClus est modifié de la façon suivante et l'algorithme sera nommé ACOClust:

Algorithme 4.8 L'algorithme AcoClust

Initialiser $\tau_{ij} = \tau_0$;
Calculer η
Initialiser au hasard la P
Par k-means sur la partition P qui converge vers un minimum local de ζ_{intra}
Pour t = 1 jusqu'à t_{max} **faire**
  Pour m = 1 jusqu'à M, **faire**
    Choisir au hasard un objet i
    Choisir un objet j avec probabilité $p_{ij}(t)$
    Affecter j à la classe de i
  FinPour
  Calculer $\zeta_{inter}(P)$ et garder la meilleure valeur
  Mettre à jour τ (t)
FinPour

b.4 Les paramètres de *ACOClust* :

La performance d'ACOClust dépend d'un certain nombre de paramètres dont les valeurs peuvent dépendre ou non des données à classifier. Tous ces paramètres ont été choisis d'une manière empirique, c'est-à-dire par tâtonnement, jusqu'à obtenir le bon résultat. Les valeurs de ces paramètres changent d'une donnée à une autre et il n'ya pas de formule pour déterminer les valeurs optimales.

Le tableau suivant résume les valeurs des paramètres d'ACOClust.

Tableau 4.2 Paramètres d'*ACOClust*

Paramètre	Description
M	Nombre de fourmis
β	contrôlent l'importance relative entre phéromones et visibilité
α	contrôlent l'importance relative entre phéromones et visibilité
ρ	un coefficient représentant l'évaporation des traces de phéromones.
T_{max}	Nombre des itérations de l'algorithme AcoClust
τ_0	Quantité de phéromone initiale
K (k-means)	Nombre de classes

4.6 Conclusion

Dans ce chapitre, nous avons présenté seulement le principe de la classification et son utilisation dans la télédétection. On a montré le lien existant entre la biomimétique et la classification. Nous avons éclairé le principe de base de la classification des images satellites. Nous avons encore présenté les différentes techniques de classification : supervisées et non supervisées.

L'ensemble des techniques de classification décrites dans ce chapitre concernent seulement ce qui est non supervisée. On a présenté les différentes techniques de classification des algorithmes de colonies de fourmis. On les a divisé en deux branches, celles qui se base sur la classification sur une grille, ou sur un tableau et finalement à base de population phéromone de fourmis.

Il est incontestable que l'utilisation des algorithmes à base de populations de fourmis a ouvert un nouveau champ pour la résolution du problème de classification (qui est rappelons le un problème NP difficile) et cela pour plusieurs raisons. D'une part, ce sont des systèmes auto-organisés et non-centralisés. Ainsi, la partition finale des données n'est obtenue que par une succession d'interactions locales entre les fourmis et l'environnement. Chaque fourmi possède un ensemble de règles comportementales qu'elle va suivre en fonction des situations qu'elle rencontre.

Les comportements de la fourmi, le tri d'objets et l'auto-assemblage ainsi la reconnaissance chimique ont poussés les ingénieurs au problème de classification de proposer plusieurs idées sur les possibilités des fourmis à résoudre le problème de la classification. Tous ces résultats étaient suite aux efforts des deux pionniers dans ce domaine : Denoubeurg et Dorigo.

Le principe de la classification des autres types de données sera considéré identique de la classification des images satellites.

Dans le prochain chapitre, l'application des algorithmes de colonies de fourmis et des techniques classiques sera présentée en deuxième partie sur les images satellites pour introduire leurs limites et par suite de citer les étapes qui nous ont conduites à la conception et la mise en œuvre des algorithmes de classification basé sur les populations de fourmis suivant notre proposition et en deuxième lieu l'application des mêmes algorithmes aux autres types de données. A la fin, une comparaison est faite avec les différents algorithmes de colonies de fourmis et k-means et une évaluation des résultats obtenus.

Chapitre 5

Implémentation et résultats

Chapitre 5 : Implémentation et résultats

5.1 Introduction

Dans ce chapitre nous allons appliquer l'ensemble des algorithmes de colonies de fourmis afin de voir leurs apports par rapports à d'autres algorithmes. Comme on l'a vu à travers les deux chapitres précédents, ces algorithmes sont divisés en deux grandes parties : l'une pour l'optimisation et l'autre pour la classification.
D'après les avancements des études sur ces algorithmes et sur leurs développements, l'idée de les appliquer est de faire d'un coté une analyse sur les résultats obtenus et de l'autre coté, là où il y a possibilité de les adapter à des types de données qui n'ont pas été traitées et de voir s'il y a encore la possibilité de les améliorer.
A travers les applications des algorithmes de colonies de fourmis et afin d'organiser notre vision, nous allons diviser ce chapitre en deux parties :

- La première traite les algorithmes de colonies de fourmis pour l'optimisation,
- La deuxième traite ces algorithmes pour la classification.

A la fin de chaque partie une analyse est effectuée suivie d'une discussion des résultats obtenus.

Nous avons utilisé la version MATLABR2010 pour effectuer les expériences de la partie optimisation et le langage de programmation C++ Builder 6.0 pour effectuer les expériences de la classification. Les caractéristiques de la machine utilisée : processeur Intel Core^{TM}i5 avec 6Go de mémoire.

5.2 Partie 1 :L'application des algorithmes à l'optimisation

5.2.1 Introduction

Notre vision à travers l'application des algorithmes de colonies de fourmis spécifique à l'optimisation se dirige à reprendre en premier lieu les algorithmes de base dans ce domaine (AS et ACO) dans un problème classique (TSP) et en deuxième lieu d'essayer d'appliquer l'algorithme ACO dans un problème réel pour voir son apport et sa possibilité de résoudre des problèmes NP-difficile.

Dans cette première partie d'optimisation, on se concentre à utiliser le premier algorithme AS pour un cas fort connue qui est le problème de TSP et dans la seconde on utilise l'algorithme ACO dans le même contexte afin de montrer que les deux algorithmes effectuent le même objectif qui est l'optimisation et que l'ACO a été élaboré afin de simplifier aux chercheurs que malgré les changements qui ont étaient faites n'a pour finalité la simplicité à l'utilisation.

Cette partie est faite sans entrer à la différence entre AS et ACO par rapport à d'autres algorithmes. Ce dernier point de comparaison sera parmi les objectifs de la seconde partie à travers l'application de plusieurs méthodes (heuristiques et méra-heuristiques) y compris l'ACO.

Seulement que la première application a été effectuée que pour un type de donnée et la deuxième avec plusieurs.

A la fin on termine chacune des deux expériences par une analyse sur l'efficacité de ces algorithmes dans la résolution des problèmes NP-difficiles.

5.2.2 Application des algorithmes Ant System (AS) et Ant Colony Optimization (ACO) pour le TSP

5.2.2.1 Application de l'algorithme Ant System (AS)

Nous allons tenter de concrétiser le fonctionnement de l'algorithme AS. Dans le cadre de ce travail, on s'est inspiré des travaux de Dorigo sur l'implémentation de AS [Dorigo,2003].

Les tests réalisés se basent sur un problème à 48 villes, le problème "Att48", qui représente une carte des grandes villes des États-Unis [Dréo, 2004].

Au début, on détermine un paramétrage qui donne de bons résultats qui est spécifié par Dréo[Dréo, 2004] :

Tableau 5.1 Paramètres d'AS d'optimalité

Paramètres	Valeurs
M	n
P	0.5
A	1 ou 2
B	5
Q	100
C	Petite valeur positive ou nulle
ρ	0.65

Si l'on choisit d'utiliser les fourmis élitistes pour améliorer la convergence, on pose e = 8. Le NC_{max} (nombre de cycles) est fixé en fonction des ressources de calcul qui sont allouées à la résolution du problème.

Avant de commencer les expériences avec l'algorithme AS Elitisme je me suis basé sur :

1- L'algorithme AS Elitisme
2- Sur les résultats des expériences effectuées par les deux chercheurs (Dorigo[Dorigo, 1996]et[Costanzo, 2006**]**) afin de s'assurer de mes résultats d'un coté et de permettre de conclure l'efficacité de l'algorithme de l'autre coté.

Il s'agissait de tester AS sur des petites instances du TSP (entre 30 et 75 villes).

Les tests réalisés se basent sur un problème à 48 villes, le problème "Att48", qui représente une carte des grandes villes des États-Unis. La solution optimale pour ce problème est présentée à la figure Fig.5.1. Elle a une longueur de 10628 km.

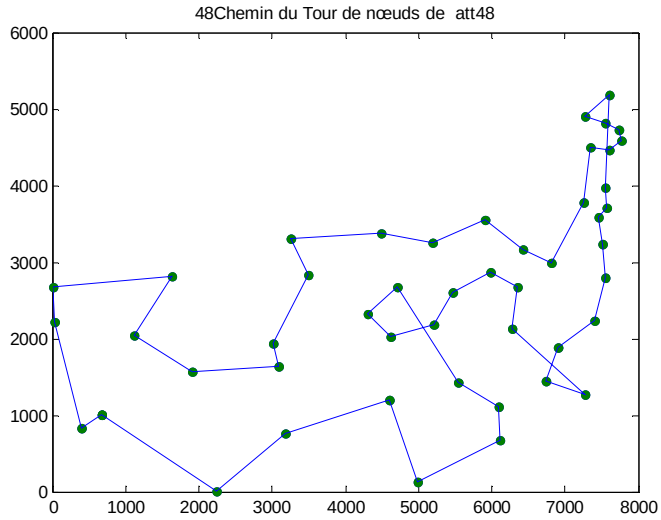


Fig. 5.1 Le problème "Att48"

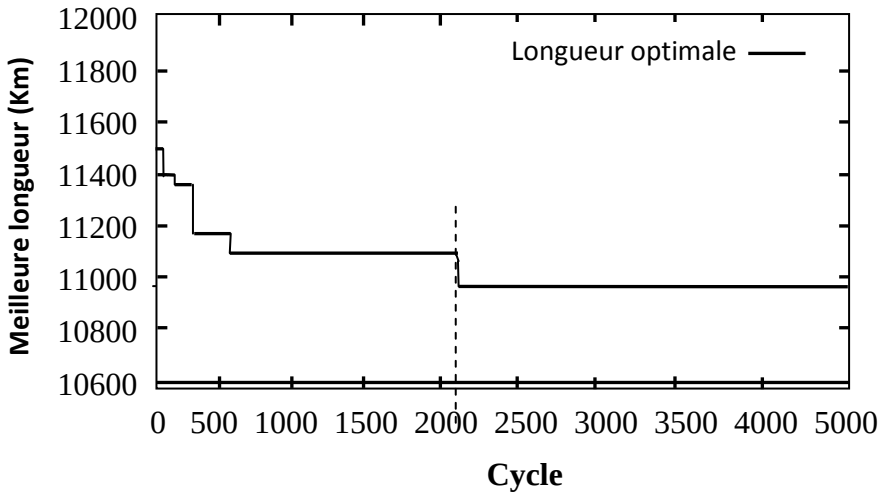


Fig. 5.2 Evolution de la longueur du meilleur tour.

5.2.2.1.1 Résultats sans fourmis élitistes

Nous avons employé pour cette expérience les paramètres définis au début de cette section le cas des sans fourmis élitistes. Nous avons laissé tourner l'algorithme durant 5000 cycles.

Le meilleur tour a été trouvé après 2139 cycles et une très bonne approximation est déjà disponible après environ 500 cycles. On peut observer l'évolution de la longueur des tours découverts à la figure Fig. 5.2. Ce tour avait une longueur de 10.957 km : sa distance relative à l'optimum était donc de (10957−10628)/10628 = 3, 095%. On remarque que AS découvre dans sa première phase de bons tours qui seront par la suite répétés.

5.2.2.1.2 Résultats avec fourmis élitistes

La même expérience a été menée avec huit fourmis élitistes e=8. On a aussi laissé courir l'algorithme sur 5000 cycles. Le meilleur tour a été trouvé au cycle 2579 à une longueur 10730 Km : ce tour a une longueur de 10730 km, soit une distance qui se rapproche du tour optimal (10730-10628)/106280=0,009597, 96% au lieu de 3, 09% évoqués plus haut).
La figure 5.3 présente cette meilleure solution.

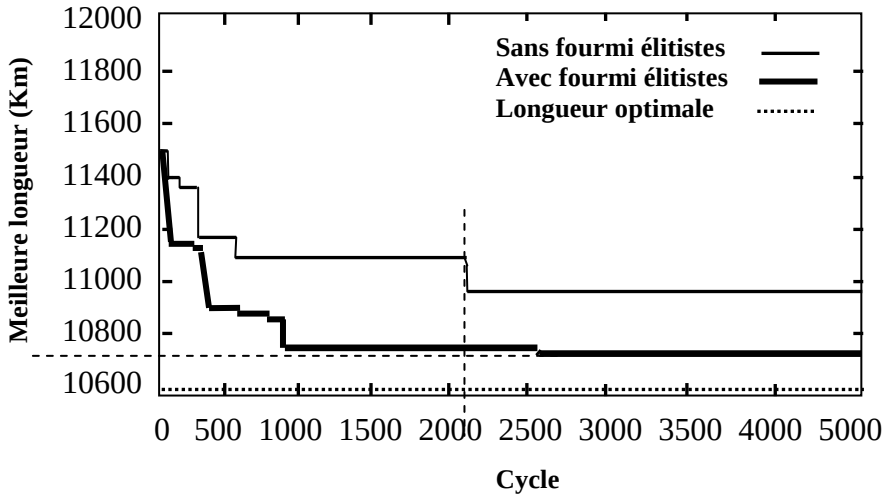


Fig. 5.3 Comparaison de la longueur du meilleur tour avec ou sans fourmis élististes.

La figure Fig.5.3 montre une comparaison de l'évolution de la longueur du meilleur tour au cours du temps selon que l'on utilise ou pas les fourmis élitistes. Grâce aux fourmis élitistes, AS converge plus vite vers une solution qui est plus proche de l'optimal.

Nous avons appliqué AS avec fourmis élitistes à d'autres problème de différentes tailles et ci-dessous le tableau récapitulatif de ces expériences :

Tableau 5.2 Résultat récapitulatif des expériences sur l'algorithme AS

Expérience	Taille du problème (nombre de villes)	Solution optimale	Longueur de la tour	Distance relative	Temps d'exécution
1	48 (non élitiste)	2139 cycles	10957 km	3, 09%	14mn 55s
2	48	2579 cycles	10730 km	0, 96%	10mn59s
3	16	1000 cycles	6867 km	9,63%	5mn12s
4	100	372 cycles	15600 km	4, 68%	25mn 13s

Exemple :

Le tour avait une longueur de 10.957 km et la **distance relative à l'optimum de la première (1) expérience est :** (10957−10628)/10628 = 3, 09%.

Remarque : Pour les expériences 1, 2 et 4 la longueur minimale est 10628 km, tandis que pour l'expérience 3 est de 646 km.

L'impact de la déficience de AS semble donc réduit sur de petits problèmes.

5.2.2.1.3 Conclusion des expériences

En guise de conclusion, il semble y avoir intérêt à exploiter les fourmis élitistes. Elles permettent d'arriver plus vite à une solution plus proche de la solution optimale. Les résultats obtenus semblent en générale identiques aux résultats des chercheurs Dorigo et Costanzo.

5.2.2.2 Application de l'algorithme ACO

Nous allons tenter de concrétiser le fonctionnement de l'algorithme ACO. Dans le cadre de ce travail, on s'est inspiré des travaux de Alhanjouri et al. sur l'implémentation d'ACO **[Alhanjouri, 2011]**.

Nous effectuons plusieurs expériences sur un jeux de données synthétiques de dimension 14 (14 nœuds) et sur trois ensembles de données réelles [AaronFoltz, 2012] avec des dimensions différentes (16, 22 et 29 dimensions), les caractéristiques de jeu de données sont illustrés dans le tableau 5.3.

Tableau 5.3 Caractéristiques des ensembles de données

Noms	Types	Commentaires	Dimensions
Ulysse16	TSP	Odyssey d'Ulysses (Groetschel/Padberg)	16
Ulysse22	TSP	Odyssey d'Ulysses (Groetschel/Padberg)	22
Bays29	TSP	29 villes à Bavaria, distances de rues(Groetschel, Juenger, Reinelt)	29
Node14	Synthetic	8 2, 0 4, -1 6, 2 -1, 4 -2, 6 0.5, 3 0, 10 3.7, 2.5, 1.8, -5 1, 7 0, 9 4, 11 3, 13 2	14

Les figures **Fig.5.4**, **Fig.5.5**, **Fig.5.6**et **Fig.5.7**montrent le chemin le plus court obtenu lorsque l'on applique l'algorithme d'ACO sur l'ensemble de données: ulysses16, ulysses22, bays29 et node14.

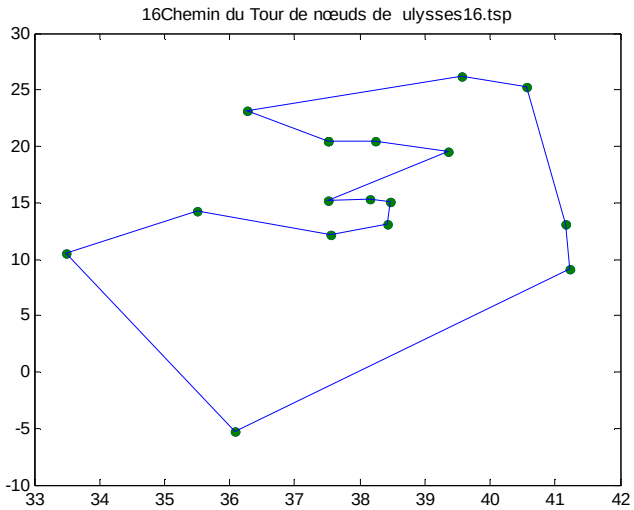


Fig.5.4.La meilleure tournée pourulysses16.tsp obtenu par l'algorithme ACO.

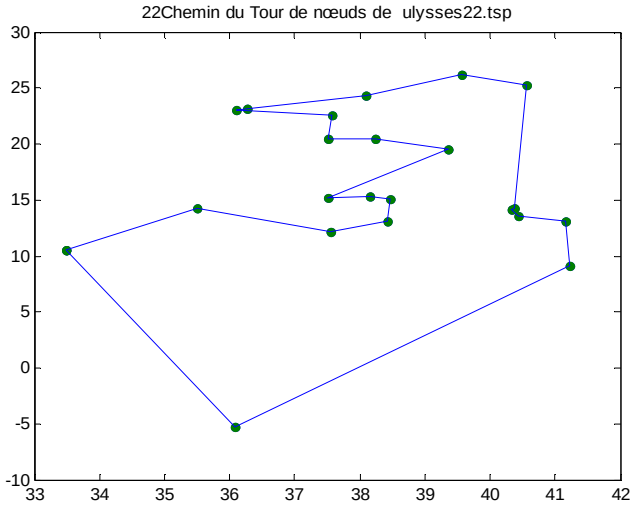


Fig.5.5. La meilleure tournée pour ulysses22.tsp obtenu par l'algorithme ACO.

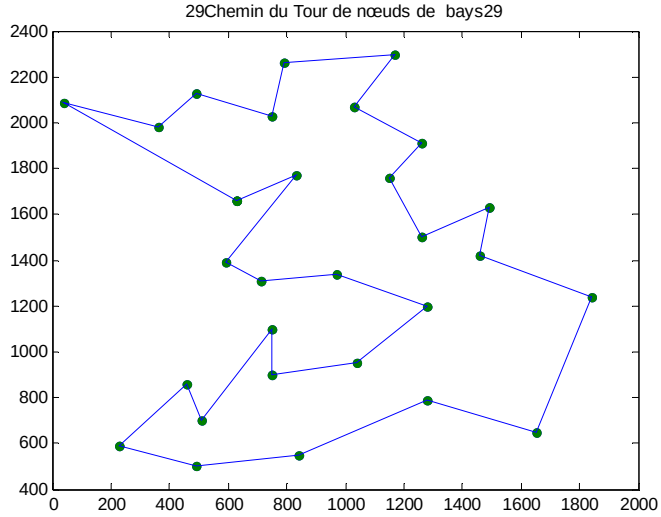


Fig. 5.6. La meilleure tournée pour bays29.tsp obtenu par l'algorithme ACO.

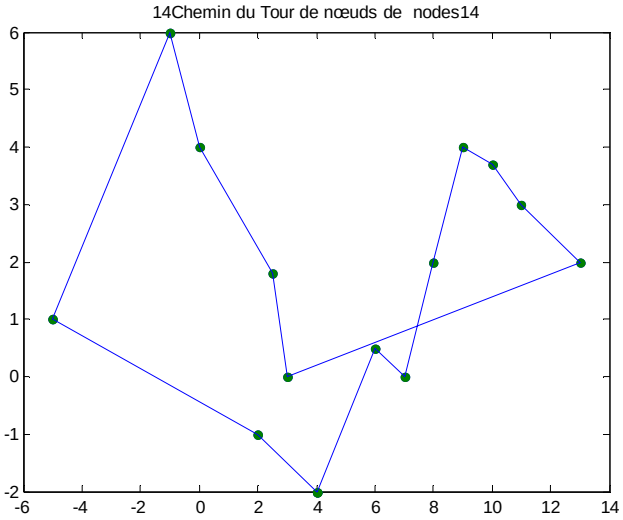


Fig. 5.7.La meilleure tournée pour node14.tsp obtenu par l'algorithme ACO.

Le tableau 5.4 présente le meilleur nombre d'itérations nécessaires pour avoir la solution optimale à laquelle nous avons la valeur minimale de la fonction objectif (le chemin le plus court).
On utilise le symbole '-'pour préciser que l'algorithme converge vers des solutions sous-optimales.
Exemple : Pour le jeu de paramètre ***ρ=.9,*** l'itération maximale est de 1400 itérations, ce qui signifie que, pour chaque jeu de paramètres de l'algorithme est exécuté dans la plagede1 à 1400itérations.Pourle jeu de paramètre ***ρ=.1,*** l'itération maximale est de 20itérations.

Ce qui signifie que si l'algorithme converge vers la valeur optimale à ***N*** itérations, nous répétons l'algorithme de plus en plus avec des itérations ***N*** pour faire en sorte que ***N*** est le nombre optimal d'itérations à paramètres ***ρ*** et ***M*** pour avoir ***f*** minimale. Trois valeurs différentes ont été utilisées (0.1, 0.5 et 0.9) pour le paramètre d'évaporation ***ρ*** et avec trois valeurs différentes (100, 250, et 500) pour le nombre de fourmis ***M***. Avec ces valeurs de paramètres on obtient les résultats présentés dans le tableau ci-dessous. Il est clair que, si le paramètre ***ρ*** est faible, alors la solution optimale est obtenue avec de petits nombres d'itérations. En outre, le nombre d'itérations est proportionnel avec le nombre de fourmis pour la résolution du problème.

Tableau 5.4 Meilleur nombre d'itérations requis par l'algorithme de convergence optimale ***f(s)***,

	ρ=.9	ρ=.5	ρ=.1	ρ=.9	ρ=.5	ρ=.1	ρ=.9	ρ=.5	ρ=.1
Base de données	M=100			M=250			M=500		
Ulysses16.tsp	-	950	130	-	-	500	-	-	200
Ulysses22.tsp	-	-	600	-	-	800	-	-	1100
Bays29.tsp	-	-	1000	-	-	-	-	-	-
Node14.tsp	1400	820	100	600	250	35	450	200	20

Le symbole '-' signifie qu'à l'itération maximale1400 l'algorithme converge vers la valeur sous-optimale, ***M***: nombre de fourmis, ***ρ***: coefficient d'évaporation.

Avec utilisation d'un grand nombre d'itérations (jusqu'à 3000 itérations) la table est en concurrence, mais avec un temps significatif (dans certains cas jusqu'à 40 min).

Tableau 5.5:Meilleur temps d'exécution requis ainsi les meilleurs fonctions optimales f(s),

Base de données	**Dimension**	**Temps d'exécution**	**Meilleur f(s)**	**Meilleur nombre d'itération**
Ulysses16.tsp	16	17s	73.987	950
Ulysses22.tsp	22	34 s	56.224	1100
Bays29.tsp	29	2 s	49.574	1000
Node14.tsp	14	1 s	45.562	1400

5.2.2.3. En conclusion,

A travers les différentes expériences on peut sortir de deux points :

- Il est clair que, si le paramètre ***ρ*** est faible, la solution optimale est obtenue avec de petits nombres d'itérations. En outre, le nombre d'itérations est proportionnel avec le nombre de fourmis pour la résolution du problème.

5.2.3 Application de l'optimisation à un problème réel

5.2.3.1 Introduction

Après avoir décrit l'application de l'algorithme AS pour résoudre de manière exacte le TSP, on s'intéresse dans cette partie à l'application de l'algorithme ACO à un problème d'optimisation réel. On abordera deux classes d'algorithmes : une méthode de résolution exact, la méthode numérique des points intérieurs, et des méta-heuristiques pour améliorer une solution.

Les méta-heuristiques suivantes : ACO, Particle Swarm optimization (**PSO**), Genetic Algorithm (**GA**) et Ant Colony Optimisation (**AS**) vont être utilisées pour résoudre un problème réel afin de comparer leurs résultats et sortir d'une conclusion finale du choix de meilleures méthodes.Ces méta-heuristiques sont récemment proposées pour les problèmes d'optimisation difficile. Notre problème d'optimisation à résoudre c'est d'optimiser une fonction objective : minimiser le coût de combustible pour la production d'énergie électrique. Ce problème traite le point suivant : le dispatching économique. La fonction objective de ce problème et les contraintes d'égalités et d'inégalités sont non linéaires.Dans notre cas on commence par l'application de la méthode de colonies de fourmis ACO. On extrait, parmi les résultats de plusieurs expériences, la meilleure solution.On effectue par suite, le même processus avec les autres algorithmes et on passe directement à la phase de comparaison.

Dans ce chapitre on va mettre en œuvre de manière pratique l'application de ces algorithmes au dispatching économique par des tests sur un réseau électrique IEEE-30 nœuds et discuter les résultats obtenus.

5.2.3.2. Formulation mathématique du problème

L'énoncé du problème consiste à minimiser une fonction objective interprétant le coût de combustible pour la production d'énergie électrique.

5.2.3.2.1. Dispatching économique (voir annexe B.3)

Les productions d'énergie électrique déterminent expérimentalement les courbes donnant le coût de production de chaque groupe en fonction de la puissance qu'il débite. La fonction associée à ces courbes est un polynôme de degré « *n* ». En pratique, le plus souvent, elle est présentée sous forme d'un polynôme du deuxième degré **[Wallach, 1986]** :

$$F_i(P_{Gi}) = c_i + b_i P_{Gi} + a_i P_{Gi}^2 \qquad \text{Eq.5.1}$$

Où : a_i, b_i et c_i sont les coefficients de coût propres à chaque unité de production.

P_{Gi} : est la puissance active générée de chaque unité de production en MW.

La minimisation de la fonction de coût total de production d'énergie électrique est une tâche qui se présente de la manière suivante :

$$Min\left\{F = \sum_{i=1}^{n} F_i(P_{Gi})\right\} \qquad \text{Eq.5.2}$$

F : Fonction objective du coût total en ***$ /h.***

F_i : Fonction coût de production de l'unité ***i***.

5.2.3.3. Application de l'algorithme ACO :

Après avoir défini les données (Voir Annexe B.3) à étudier on passe directement à appliquer l'algorithme ACO qui est adapté à ce type de problème.
La difficulté à l'application de l'algorithme ACO réside dans le choix des paramètres.
Nous avons généré aléatoirement des tableaux contenant seulement les résultats optimaux de chaque expérience.

Manipulation du paramètre d'évaporation

On fixe α, β et le nombre de fourmis et on fait varier ρ (paramètre d'évaporation) de 0.4 avec pas de 0.1 jusqu'à 0.9 et le nombre de fourmis = 10.

- α=1, β=5, NbCycle=100

Tableau 5.6 Expériences d'évaluation des coûts de production

Puissances génères [MW]	**Pour chaque ρ on extrait les P_{Gi} de chaque meilleur coût de production**					
	ρ = 0.4	**ρ = 0.5**	**ρ = 0.6**	**ρ = 0.7**	**ρ = 0.8**	**ρ = 0.9**
P_{G1}	50,5423	74,302	64,6591	61,3564	47,996	64,6414
P_{G2}	32,408	62,934	22,706	77,271	35,992	62,70663
P_{G3}	69,575	113,888	66,885	94,7499	29,939	97,658
Coût de production[$ /h]	**21044,76**	**24213,04**	**24127,37**	**25340,95**	**25749,41**	**23818,73**

Bien que la variation du paramètre d'évaporation, on remarque la difficulté de décider du meilleur coût à travers un ρ (paramètre d'évaporation) spécifique.
Malgré tous cela, on a refait plusieurs expériences avec une variation du ρ (paramètre d'évaporation) et on le fait varier entre [0 et 10].

Tableau 5.7 Coûts de production de différents paramètres d'évaporation

ρ (paramètre d'évaporation)	**Coût de production [$ /h]**
0,1	**25543**
0,6	**24127,37**
1,1	**23846**
1,6	**24582,40**
2,1	**24582,62**
2,6	**21345**
3,1	**20755**
3,6	**21689,65**
4,1	**23558,93**
4,6	**23842**
5,1	**24385,78**

5,6	**24832,24**
6,1	**24912**
6,6	**24147,04**
7,1	**235271**
7,6	**24698**
8,1	**23435,04**
8,6	**22319**
9,1	**23543**
9,6	**25536**
10,1	**23845**

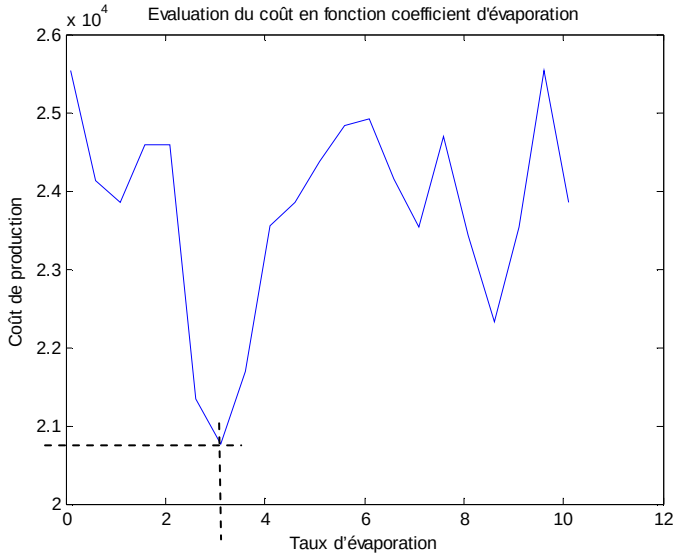


Fig. 5.8 Evaluation du coût de production en fonction du coefficient d'évaporation.

Ce qui nous pousse à dire que pour **ρ** =3.1 on aura une valeur minimale du coût de production à **20755** [$ /h] et ceci est clair dans le tableau ci-dessus.

Tableau 5.8 Tableau des paramètres de l'algorithme ACO optimaux :

α	**β**	**NbCycle**	**ρ**	**Nfourmi**
1	5	100	3.1	10

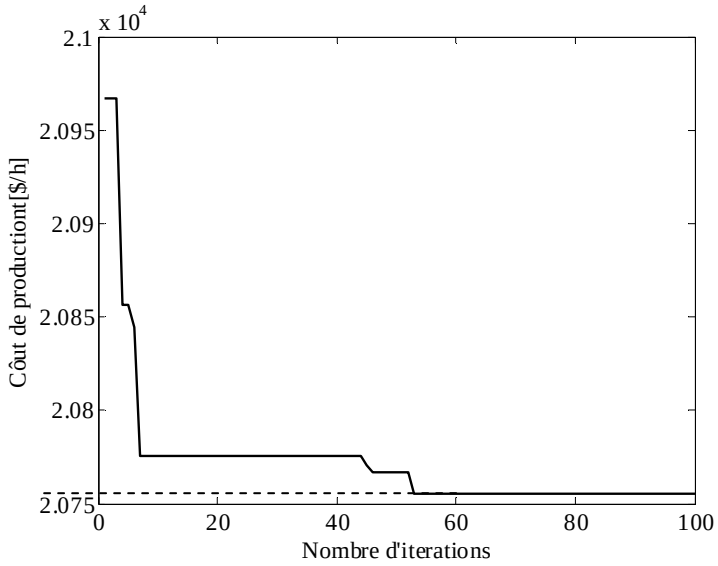


Fig. 5.9 Variation du coût de production en fonction du nombre d''itération.

Tableau 5.9 Le coût de production et les puissances générées optimaux obtenus par ACO

Puissances optimales générées [MW]			
puissances optimales générées [MW]	P_{G1}=78.3125	P_{G2}=174.5658	P_{G3}=152.1213
Coût de production optimal [$/h]	**20755**		

5.2.3.4. Application de l'algorithme génétique

Le problème est le même que celui cité en ACO ci-dessus.

Les paramètres de l'algorithme génétique optimaux sont donnés par le tableau 5.10 :

Tableau 5.10 Les paramètres de l'algorithme génétique optimaux

Nombre de population initial	**Probabilité de croisement**	**Probabilité de mutation**
100	0.6	0.05

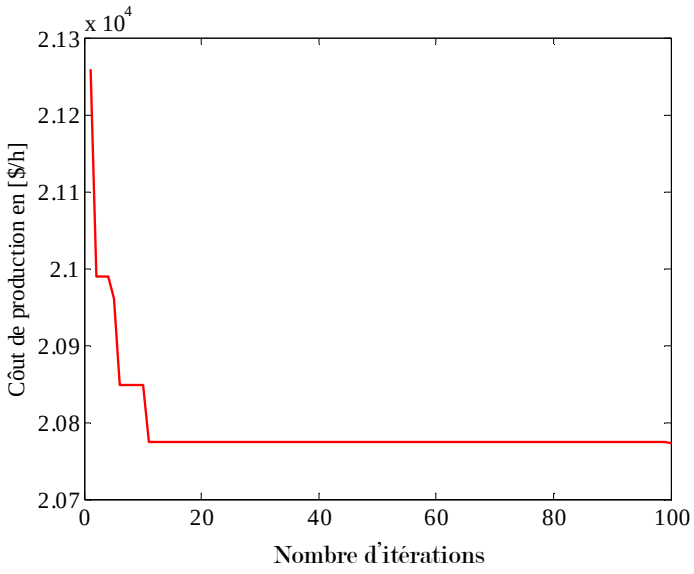


Fig. 5.10. Variation du coût de production en fonction du nombre d'itération. De l'algorithme génétique.

Tableau 5.11 Le coût de production et les puissances générées optimaux obtenus par l'algorithme génétique.

Puissances optimales générées [MW]			
puissances optimales générées [MW]	P_{G1}=89.2308	P_{G2}=160.2857	P_{G3}=156.3651
Coût de production optimal [$/h]	**20773**		

5.2.3.5 Application de l'algorithme Parallel Asynchronous PSO

Le problème est le même que celui cité en ACO.

Tableau 5.12 Tableau des paramètres de l'algorithme PAPSO optimaux :

Nombre de particule	**Wmax**	**Wmin**	**c1=c2**
30	0.6	0.1	1.5

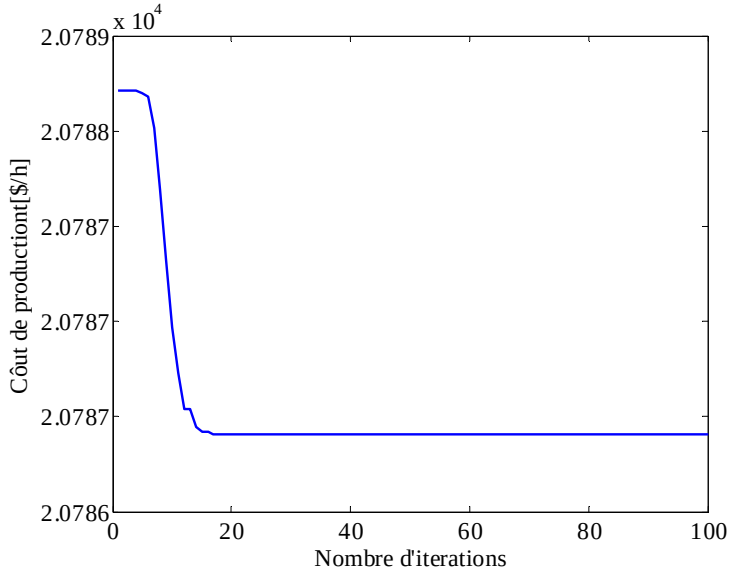


Fig. 5.11. Variation du coût de production en fonction du nombre d'itération de l'algorithme PAPSO.

Tableau 5.13 Le coût de production et les puissances générées optimaux obtenus par l'algorithme PAPSO.

Puissances optimales générées [MW]			
puissances optimales générées [MW]	P_{G1}=77.3125	P_{G2}=176.5658	P_{G3}=153.1213
Coût de production optimal [$/h]	**20786**		

5.2.3.6. Application de l'algorithme Point intérieur

Le problème est le même que celui cité en ACO ci-dessus.

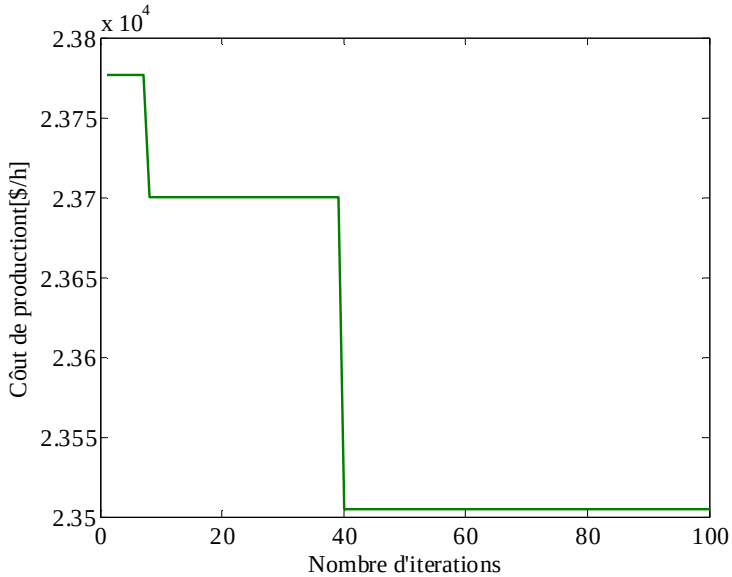


Fig. 5.12. Variation du coût de production en fonction du nombre d'itération de l'algorithme point intérieur.

Tableau 5.14 Le coût de production et les puissances générées optimaux obtenus par l'algorithme point intérieur.

Puissances optimales générées [MW]			
puissances optimales générées [MW]	P_{G1}=72.3791	P_{G2}=180.1811	P_{G3}=153.4398
Coût de production optimal [$/h]	**23505**		

5.2.3.7. Récapitulation de l'application des méthodes sur le réseau test IEEE-30 nœuds

Pour une transmission de 400 MW et des pertes de transmission est de 7MW, les résultats des puissances actives et du coût total de production sont donnés dans le tableau suivant :

Tableau 5.15 Tableau récapitulatif des coûts de production et les puissances générées optimaux obtenus par les algorithmes ACO, Génétique, PAPSO et point intérieur.

		Algorithmes d'optimisations			
		ACO	Génétique	PAPSO	Point intérieur
Puissance optimales [MW]	**P_{G1}**	78.3125	89.2308	77.3125	72.3791
	P_{G2}	174.5658	160.2857	176.56	180.1811
	P_{G3}	152.1213	156.3651	153.1213	153.4398
	Coût de production $/h	**20755**	**20773**	**20786**	**23505**

Le tableau 5.15 montre les valeurs de puissances actives optimales, et le coût de production optimale. On remarque que tous les résultats sont très proches sauf pour le point intérieur. La valeur moyenne du coût pour les quatre méthodes d'optimisations est de l'ordre de 20777 $/h.

La valeur minimale du coût est de **20755 $/h,** obtenue par l'ACO, correspond à la combinaison (β = 5, ρ = 3,1, α= 1, Nfourmi=10 et NbCycle=100).

Les puissances actives optimales sont dans leurs gammes permises et sont loin des limites minimum et maximum.

Ainsi, les résultats de nos expériences réalisés sont proche des résultats de plusieurs chercheurs **[Benasla, 2008] [Abderrahmani, 2011]** (PAPSO, génétique) et finalement notre apport avec l'ACO donne des meilleurs coûts que les différents algorithmes utilisés.

Le tableau 5.16 montre l'amélioration du coût de production obtenue par l'application de l'ACO par rapport aux autres méthodes.

Tableau 5.16 Amélioration du coût de production de l'ACO par rapport algorithme génétique, PAPSO et point intérieur.

Amélioration du coût de production de l'ACO par rapport algorithme génétique	**18 [$/h]**
Amélioration du coût de production de l'ACO par rapport PAPSO	**31 [$/h]**
Amélioration du coût de production de l'ACO par rapport point intérieur	**2750 [$/h]**

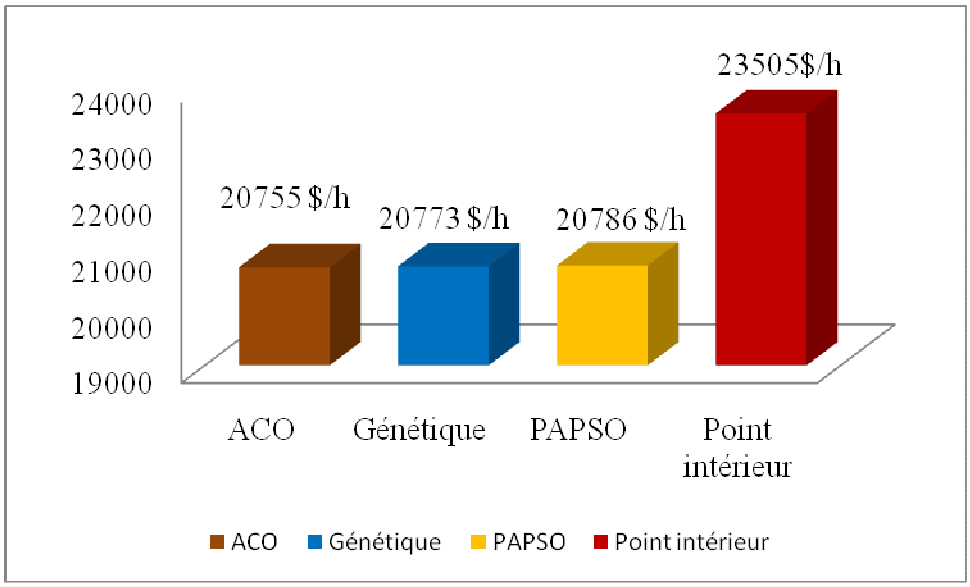


Fig. 5.13.Coût de production pour les différentes méthodes.

5.2.3.8. Conclusion

Dans cette partie d'utilisation de l'algorithme de fourmis pour un cas réel, on a testé la méthode ACO pour l'optimisation du coût de combustible d'un réseau test IEEE-30 nœuds. Le problème a été formulé en tant que mono-objectif pour optimiser le coût de combustible. Les résultats obtenus sont satisfaisants comparés avec ceux trouvés par les méta-heuristiques (PAPSO et AG) et la méthode heuristique (point intérieure). L'algorithme ACO nous a permis de trouver des résultats satisfaisants. Il est beaucoup plus efficace et performant dans ces types de problèmes.

Le problème du dispatching économique a déterminé les niveaux de production de l'ensemble des générateurs pour garantir l'équilibre entre la production et la consommation au moindre coût.

Le développement rapide de la technique du calcul numérique a permis de maîtriser ce problème en élaborant plusieurs algorithmes de calcul permettant de savoir combien d'énergie faut-t-il générer par les groupes et combien d'énergie faut-t-il transférer à travers chaque nœud afin de répondre aux exigences des consommateurs avec un minimum de coût de production.

Ces algorithmes méta-heuristiques utilisent une recherche stochastique aléatoire. Nos résultats de simulation montrent l'efficacité de ces algorithmes dans l'étude d'un dispatching économique.

5.3 Partie 2 : L'application des algorithmes à la classification

5.3.1. Introduction

On s'intéresse dans ce chapitre à appliquer les algorithmes de colonies de fourmis pour la classification sur plusieurs types de données.

En premier lieu, on traite les images satellites et en deuxième lieu de quelques données de Machine Learning Repository (Voir Annexe B), afin d'arriver en dernier lieu à sortir d'une comparaison des algorithmes de colonies de fourmis et d'une évaluation des résultats obtenus.

Dans ce chapitre, on commence à appliquer deux algorithmes de classification non supervisées qui sont k-means et ISODATA puis nous nous intéressons à l'application des algorithmes de colonies de fourmis et en particulier les algorithmes AntClust et ACOClust.

Ces deux algorithmes sont étudiés, adaptés et appliqués au problème de la classification non supervisée des données choisies, afin de montrer leurs efficacités de traitement des données de différentes natures.

5.3.2. Zone d'étude et données satellites traitées [benya, 2008] :

Nous allons appliquer des classifications multispectrales utilisant les différentes approches non supervisées sur une image Thématic Mapper prise par le satellite LANDSAT 5 le 15 Mars 1993 à 9h45, la région d'Oran ouest. (Voir Annexe B).

Les tests ont été faits sur une fenêtre de taille 700 lignes x 800 colonnes de la scène prise par le satellite LANDSAT 5.

Dans notre travail, nous avons utilisé l'image satellite, citée en annexe B, existante dans le laboratoire de télédétection où j'ai effectué mes expériences et que toutes les étapes de prétraitements et de rehaussement ont été fait sur cette dernière.

Mais, avant d'entamer tous ce qui a été dit, on présente les tests effectués sur les deux approches classificatoires non supervisées (k-means et ISODATA) avec différents nombres d'itérations, pour pouvoir comparer les résultats obtenus d'une part, évaluer l'efficacité de ces algorithmes d'autre part, et afin de choisir la méthode qui va être utilisée durant la phase de comparaison. L'évaluation s'effectue par des tests de validation à partir des échantillons choisis sur l'image initiale correspondant aux thèmes identifiés sur l'image.

5.3.3 Classification par approche d'Isodata et k-means (voir annexe C)

5.3.3.1 Classification par approche d'Isodata

<u>Les paramètres :</u>

Nous avons appliqué cette approche sur les données TM avec les paramètres suivants :

- Le nombre de classes : Le nombre de classes identifiées sur l'image est de 11. Comme la classe Ressac est totalement confondu avec la classe Mer, après la classification, on a choisi un nombre égal à 10.

- Le nombre d'itérations : l'application de cet algorithme avec une itération ne donne pas un résultat satisfaisant, l'image obtenue comporte des confusions entre différents thèmes, ces confusions sont moins importantes à partir de la 3ème itération.

- Distance minimale entre les classes : La distance minimale entre deux classes est fixée à 10 pixels dans le but de bien les disperser.

Les images résultats obtenues après l'application de cette approche avec un nombre d'itération égal à 3, 6, 9 et 12 sont montrées dans la figure (Fig. 5.14) :

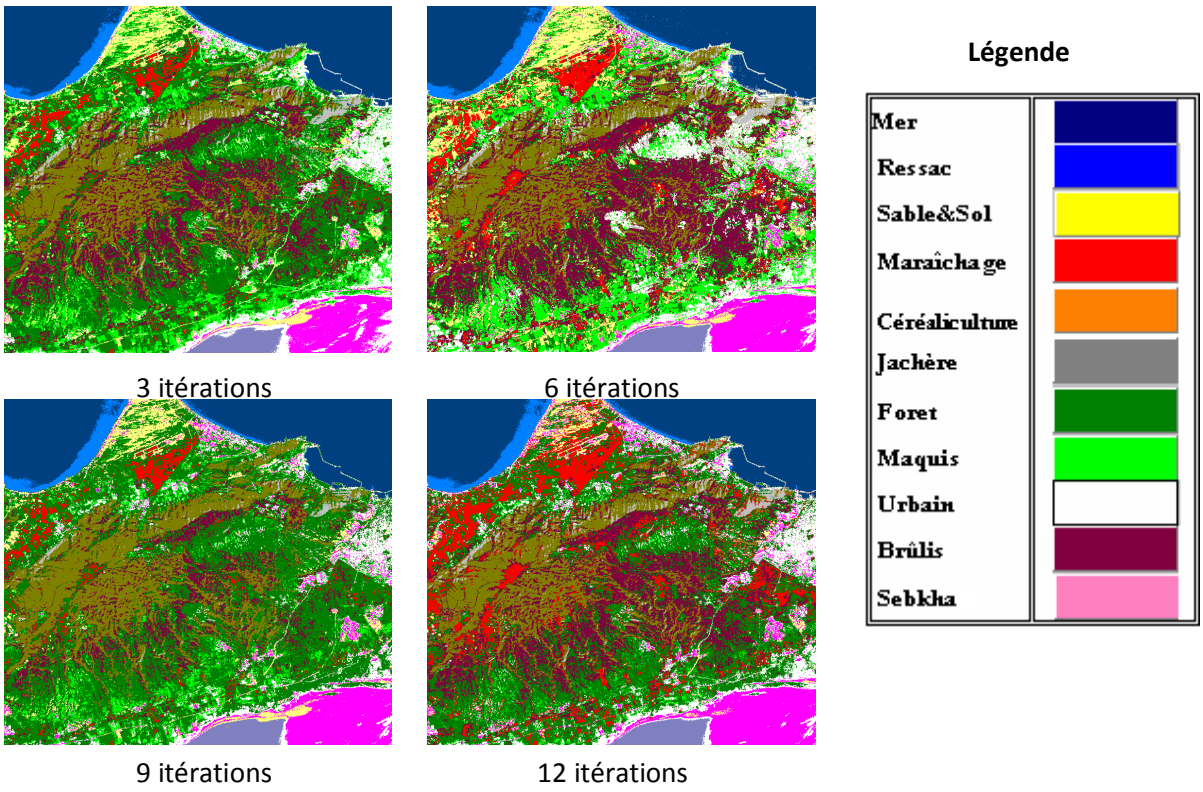


Fig. 5.14 Résultat de la classification par l'approche d'Isodata

5.3.3.2 Classification par approche du K-means (voir annexe C)

Les paramètres :

Les paramètres appliqués pour cette méthode sont :

- Le nombre de classes : est fixé à 11 et correspondant aux thèmes présents sur l'image initiale.

- Le nombre d'itérations : comme pour l'approche d'Isodata, le nombre d'itération minimal qui nous permet de distinguer entre les thèmes sans grande difficulté est de 3.

La figure ci-dessous (Fig.5.15) représente les images résultats obtenues après application de l'algorithme K-means avec 3, 6, 9 et 12 itérations.

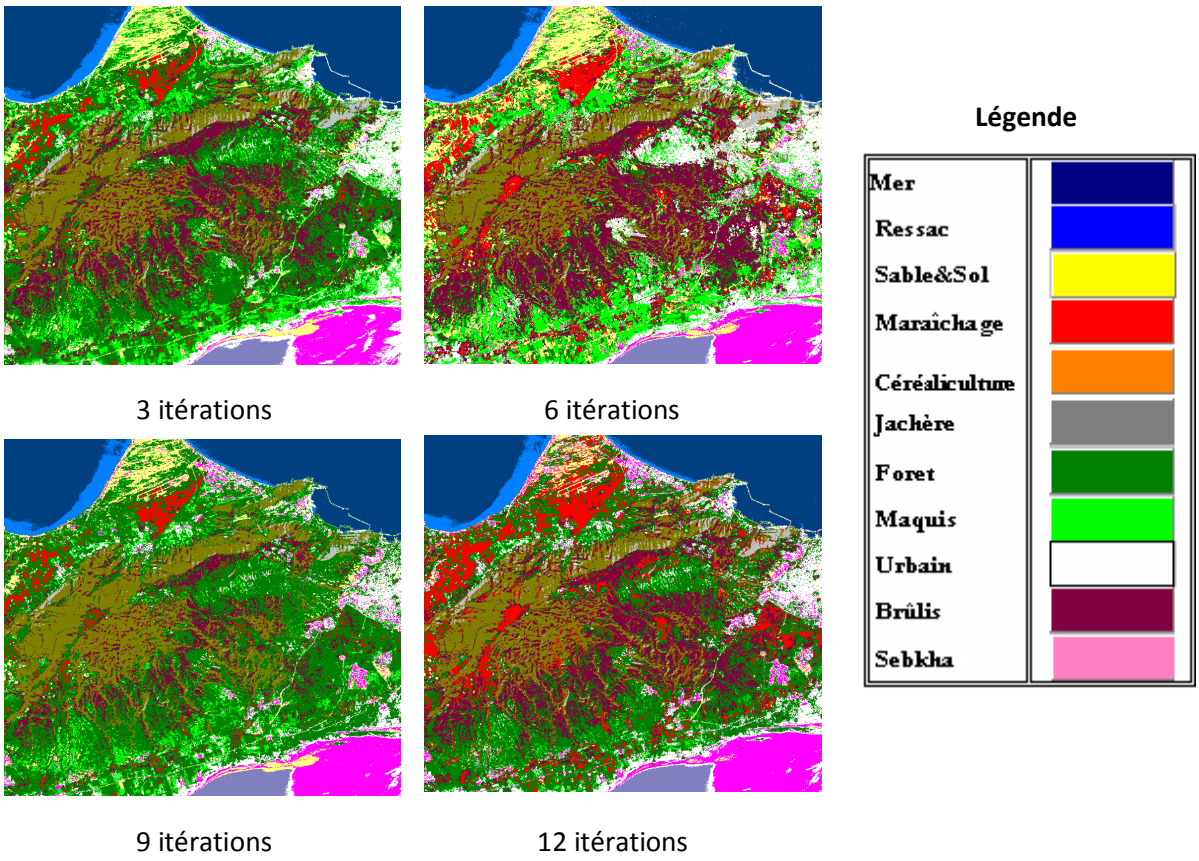


Fig. 5.15 Résultat de la classification par l'approche K-means

5.3.3.3 Résultats et interprétation :

Pour évaluer la performance des deux approches utilisées, on a procédé à des tests de validation pour 3, 6, 9 et 12 itérations. Pour un nombre d'itérations égal à 1 et 2, l'image résultante contient des confusions importantes entre différentes classes. Ces confusions sont faciles à détecter visuellement et à être confirmées par les tests de validations. Ces confusions ne sont diminuées qu'à partir de la troisième itération. Au-delà de la 12ème itération, les résultats obtenus restent inchangés. On a choisit le premier groupe d'échantillons pour évaluer chaque méthode. La précision totale de chaque classification a été calculée à partir de la matrice de validation correspondante. Pour dresser cette matrice, on a fait la correspondance entre chaque classe obtenue par la classification et les classes réelles de l'image. La précision totale relative à chacune des deux méthodes et pour chaque nombre d'itérations est définie dans le tableau suivant :

Tableau 5.17 Précision totale des deux approches classificatoires

	3 itérations	6 itérations	9 itérations	12 itérations
K-means	41.953%	51.084%	53.47%	53.74%
Isodata	42.89%	41.12%	43.45%	43.45%

Le graphe suivant schématise les résultats du tableau :

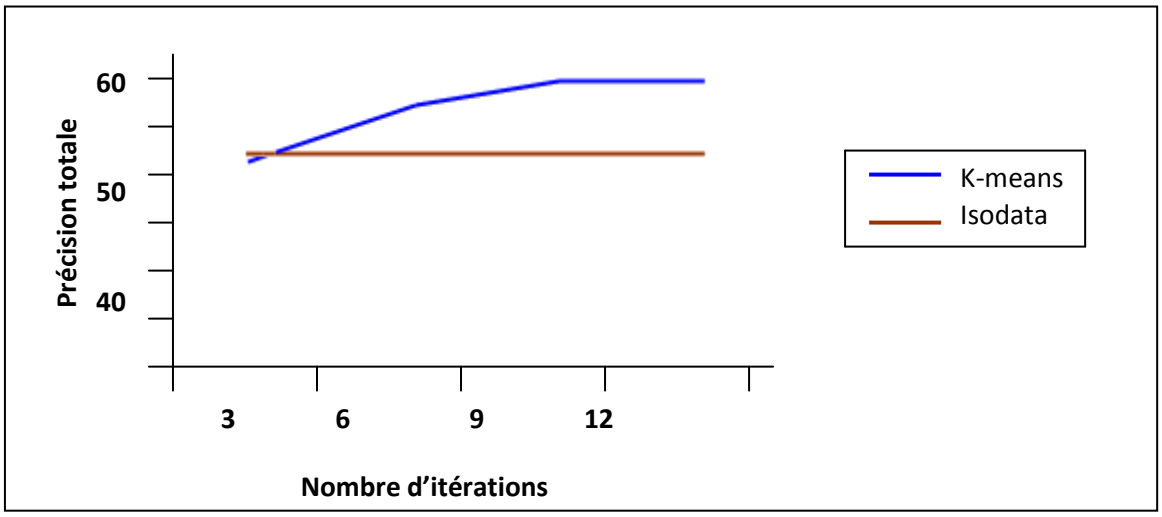


Fig. 5.16 Graphe comparatif entre les deux approches classificatoires (K-means & Isodata)

A partir de ces résultats, on peut soulever les points suivants :

- En augmentant le nombre d'itération, la classification devient de plus en plus précise et par conséquent, l'image résultante comportera moins de confusion entre les thèmes.

- La classification par approche du K-means donne de meilleurs résultats que ceux obtenus par l'approche d'Isodata.

5.3.4. Etude expérimentale de l'algorithme AntClust

Dans cette partie on passe directement à l'application de l'algorithme AntClust où la colonie de fourmis est disposée suivant une loi aléatoire uniforme sur l'image à classifier.

Cette dernière est assimilée à un tableau, et ceci après abandonné la grille sur lequel se déplaceront les fourmis.

On ne va pas reprendre la formulation de l'algorithme AntClust qui est citée précédemment dans le chapitre 4.

Les opérations de ramassage et de dépôt des pixels ne sont pas effectuées d'une manière physique mais virtuellement suivant des mesures de probabilités calculées dans le voisinage spatial du pixel sur lequel la fourmi est déposée. La mesure de similarité appliquée entre deux pixels est la distance euclidienne radiométrique.

Les paramètres de ramassage et de dépôt $\boldsymbol{P_d}$ et $\boldsymbol{P_p}$ considérés sont compris entre 0 et 1 et qui permettent d'obtenir des classes plus homogènes, ce qui limite les erreurs de partitionnement. Ces deux paramètres sont déterminés d'une manière empirique et la capacité de transport d'une fourmi est égale à 1. Cette première étape est un processus itératif qui s'achève lorsque le taux d'exploration des pixels est de 100% ou égal à un certain seuil. A la fin de cette étape, une image étiquetée est obtenue.

Les différents paramètres proposés suivent les règles suivantes:

- Le nombre de fourmis à choisir dans l'algorithme AntClust est proportionnel au nombre de pixel à classer. Un nombre important de fourmis peut assurer un taux d'exploration de 100% de l'image (absence de pixels libres), mais entraînera un temps de calcul très élevé. De ce fait, on a recours à un nombre raisonnable de fourmis. (Dans notre application aucune possibilité de choisir un nombre supérieur à 500),
- Un nombre élevé des itérations (le nombre maximum possible). Dans ce cas, on a obligé de choisir un nombre acceptable.
- Les autres paramètres : Sont des valeurs qui ont été définies par des expériences des chercheurs du domaine [Ouadfel, 2006].

Nous avons fixé P_p = 0.015 et P_d = 0.56, qui sont des paramètres empiriques **[Khedam,2008]**, avec **50** fourmis, le nombre d'itérations $\boldsymbol{t_{max}}$=$\mathbf{3*10^6}$, le paramètre de contrôle de la probabilité de déposer un pixel dans une case $\boldsymbol{P_p}$ = 0.1, le paramètre de contrôle de la probabilité de choisir un pixel dans une case contenant deux pixels q = 0.7 et le paramètre de contrôle de dilatation de la fonction de similarité β = 50.

L'image est assimilée au tableau sur lequel se déplaceront les fourmis. Les opérations de ramassage et de dépôt des pixels ne sont pas effectuées d'une manière physique mais virtuellement suivant des mesures de probabilités calculées dans le voisinage spatial du pixel sur lequel la fourmi est déposée. La mesure de similarité appliquée entre deux pixels p_i et p_j est la distance euclidienne des niveaux de gris, ng_i et ng_j, comprise entre 0 et 1 (Eq. 4.5).

Les paramètres de ramassage et de dépôt $\boldsymbol{P_d}$ et $\boldsymbol{P_p}$ considérés sont compris entre 0 et 1 et qui permettent d'obtenir des classes plus homogènes, ce qui limite les erreurs de partitionnement. Ces deux paramètres sont déterminés d'une manière empirique et la capacité de transport d'une fourmi est égale à 1. L'algorithme AntClust est un processus itératif qui s'achève lorsque le taux d'exploration des pixels est de 100% ou égal à un certain seuil.

L'algorithme AntClust est effectué où la colonie de fourmis est disposée suivant une loi aléatoire uniforme sur l'image à classifier.

A la fin, une image étiquetée est obtenue.

Cependant, cette image risque de présenter deux problèmes:

1. Un nombre élevé de classes **(Fig.5.17)**,
2. Le déplacement des fourmis étant aléatoire, il y a risque de présence de pixels non classifiés (pixels libres). Pour une résolution de ces problèmes, on propose une deuxième phase d'exploration durant laquelle un autre algorithme à base de fourmis est appliqué.

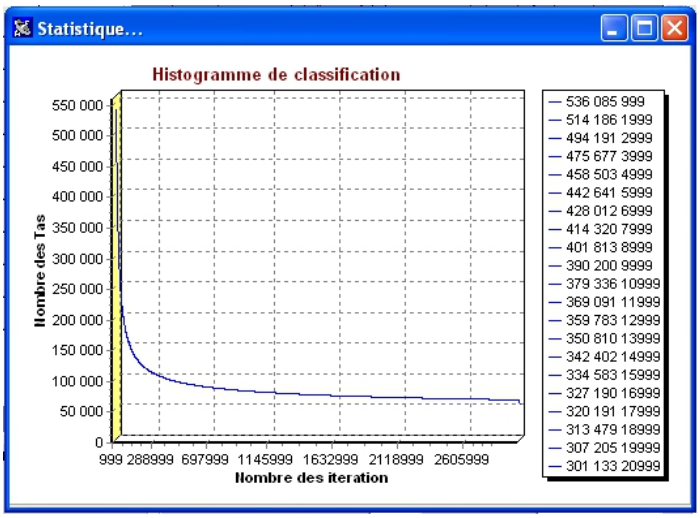


Fig. 5.17 Évolution du nombre des tas au cours des itérations

Suivant la figure (Fig.5.17) on constate, après un nombre d'itération t_{max}=**3*10^6**, que l'algorithme converge vers un nombre énorme (**50 000**) de classes (Tas) et il est loin de la réalité. Plusieurs expériences ont été effectuées avec d'autres nombres d'itérations 3.5*10^6 et 4*10^6 et le résultat reste inchangé (le nombre de tas converge toujours au nombre 50 000). Ce qui oblige à revoir de plus près l'algorithme et de faire des adaptations à cet algorithme afin d'arriver à réduire le nombre de classes.

5.3.4.1. L'algorithme AntClust adapté

L'algorithme AntClust n'est pas optimal et il est incapable d'estimer le nombre de classes. L'idée consiste à considérer les petites classes homogènes comme des objets eux-mêmes, on augmente en quelque sorte la capacité de transport des fourmis. Afin de traiter les objets par les fourmis, nous avons adapté l'algorithme AntClust et considérer que les fourmis peuvent transporter un Tas entier d'objets.

En effet, l'algorithme AntClust adapté aux pixels d'une image satellite suit les deux étapes suivantes :

A. Première étape (AntClust stochastique)

La première étape consiste en une application directe de l'algorithme AntClust (AntClust stochastique) décrit précédemment. Les règles et les heuristiques qui simulent le comportement naturel des fourmis restent sensiblement les mêmes, les principales modifications se situent au niveau des points suivants :

- La génération du tableau spécifié initialement par AntClust entraîne un temps de traitement très important. Pour notre application, et dans le but de réduire le temps de calcul, l'image multispectrale à classifier est assimilée à un tableau sur lequel se déplaceront les fourmis. Sa taille est donc déterminée automatiquement en fonction des dimensions de cette image,
- Les deux paramètres de classification P_p et P_d sont déterminés d'une manière empirique relativement à l'image à classifier,
- La capacité de transport des fourmis est égale à 1.

La meilleure solution serait d'adapter AntClust aux deux problèmes précédents. Pour cela, nous avons effectué une seconde étape de classification en utilisant toujours les fourmis.

B. Deuxième étape :(AntClust déterministe)

L'attribution d'une classe à chaque pixel et la réduction du nombre de classe s'effectue suivant les étapes suivantes :

1- La colonie se compose d'une seule fourmi,

2- Le déplacement de la fourmi est déterministe,

3- La fourmi dispose d'une mémoire interne propre pour cibler directement les pixels libres,

4- La capacité de transport de la fourmi est infinie, la fourmi devient capable de manipuler des tas d'objets,

5- Les règles heuristiques de base sont adaptées pour permettre à la fourmi de manipuler des tas entiers,

6- Les paramètres $\boldsymbol{P_d}$ et $\boldsymbol{P_p}$ sont aussi utilisés.

Doter la fourmi de la faculté de la mémoire permet de cibler directement les pixels libres et les Tas d'objets à classifier, d'accélérer la classification et d'accroître la proportion des objets qui seront affectés à un Tas. Les comportements de dépôt et de mouvement de la fourmi sont modifiés de la manière suivante : quand une fourmi transporte un objet (ou un tas d'objets), elle cherche dans sa mémoire un tas T_j sur lequel elle pourrait déposer l'objet (ou le tas). Si elle en trouve un, alors elle se dirige vers lui. Si la fourmi n'a pas déposé son objet en chemin, elle le déposera sur T_j s'il est toujours valable, c'est-à-dire s'il n'a pas été détruit ou s'il n'a pas été trop modifié par d'autres fourmis.

En récapitulatif, supposons que AntClust stochastique a généré une image étiquetée contenant K tas d'objets (K classes) et m pixels libres. AntClust déterministe opère de la manière suivante :

- Pour classer les m pixels libres, la fourmi utilise sa mémoire interne pour aller directement à ces pixels afin de leur attribuer une classe parmi les K classes initiales;
- Pour réduire le nombre de classes K, la fourmi sera capable de transporter un Tas entier de pixels (capacité infinie). $P_c(T)$ est le centre de gravité du Tas T et $d^*(T)$ est la distance maximale entre deux pixels de T. La fourmi dépose un Tas T_1 sur un autre Tas T_2 si

$$\frac{d(p_c(T_1), p_c(T_2))}{d^*(T)} \leq P_d \quad \text{(voir Eq.4.6)}$$

Quand T_1 et T_2 sont rassemblés, ils forment un unique Tas T_3 . Cela signifie que deux tas qui ont été rassemblés ne peuvent plus être séparés. Ceci accélère la convergence de AntClust déterministe. La convergence est atteinte lorsque le taux de changement des classes des pixels est faible ou nul entre deux itérations consécutives.

Pour réduire le nombre de Tas, la fourmi sera capable de transporter un Tas d'objets. La fourmi dépose un Tas T_1 sur un Tas T_2 si la distance entre les centres de gravités des deux Tas est inférieur à $\boldsymbol{P_p}$ (c'est une dissimilarité maximale autorisée pour créer un Tas de deux Tas existants et son intervalle est entre 0.05 et 0.2) (Voir algorithme 2 : algorithme déterministe);

Quand T_1 et T_2 sont rassemblés, ils forment un unique Tas T_3. Cela signifie que deux Tas qui ont été rassemblés ne peuvent plus être séparés. Ceci accélère la convergence d'AntClust de cette deuxième étape. La convergence est atteinte lorsque le taux de changement des classes des pixels est faible ou nul entre deux itérations consécutives.

Remarques

1. Le nombre de pixels libres est inversement proportionnel au nombre de fourmis mis en jeu dans l'algorithme.

2. Le nombre de fourmis à choisir dans l'algorithme AntClust est proportionnel au nombre d'objets à classer. Un nombre important de fourmis peut assurer un taux d'exploration de 100% de l'image (absence de pixels libres), mais entraînera un temps de calcul très élevé. De ce fait, on a recours à un nombre raisonnable de fourmis et à la deuxième étape de l'algorithme.

3. Dans l'algorithme AntClust Stochastique, les classes sont découvertes automatiquement de la manière suivante :

- Lorsqu'une fourmi ne transporte pas de pixels, elle vérifie son voisinage, deux cas se présentent :

a. Si ce voisinage n'est pas étiqueté, elle ramasse le pixel le plus dissimilaire de ce voisinage et forme par le reste des pixels une nouvelle classe qui portera une nouvelle étiquette.

b. Si ce voisinage est déjà étiqueté, la fourmi ramasse l'objet le plus dissimilaire de ce voisinage, les pixels qui restent gardent la même étiquette (pas de nouvelle étiquette).

- Lorsqu'une fourmi transporte un pixel, elle vérifie d'abord son voisinage, deux cas se présentent :

a. Si aucun tas (classe) ne se trouve dans le voisinage immédiat de la fourmi (aucune autre fourmi n'a encore visité ce voisinage, pixels non étiquetés) et si le pixel qu'elle transporte peut être regroupé avec les pixels non étiquetés, il y a création d'une nouvelle classe à qui on associe une nouvelle étiquette. Par exemple, soit le nombre de classes déjà formées égale à 0. Quand la fourmi dépose le pixel, il y a création d'une première classe qui portera l'étiquette 1.

b. Si par contre, la fourmi se trouve en face d'un voisinage déjà étiqueté (par la même fourmi ou une par autre) et que le pixel qu'elle transporte peut être déposé sur ce tas, on attribue alors à ce pixel l'étiquette de ce tas (pas de nouvelle étiquette).

Ce premier processus est suivi directement par la deuxième étape (AntClust déterministe) décrit précédemment.

La complexité d'AntClust stochastique est en *O(TM)* : tels que T est le nombre d'itération maximale et M est le nombre de fourmi. La complexité de l'algorithme AntClust déterministe est difficile à déterminer car il se base sur les tas trouvés dans AntClust stochastique ce qui nous permet de confirmer que le nombre de tas est inférieur au nombre d'itération T et en plus on a seulement qu'une fourmi alors la complexité du deuxième algorithme est inclus à la complexité de AntClust stochastique.

Complexité générale d'AntClust est : ***O(TM)***

5.3.5 Etude expérimentale de l'algorithme ACOClust

Les paramètres proposés avec des règles :

- Le nombre de fourmis à choisir dans l'algorithme ACOClust est proportionnel au nombre de pixel à classer. Un nombre important de fourmis peut assurer un taux d'exploration de 100% de l'image (absence de pixels libres), mais entraînera un temps de calcul très élevé. De ce fait, on a recours à un nombre raisonnable de fourmis (entre 200 et 300 fourmis).
- Un nombre élevé des itérations (le nombre maximum possible).
- L'application de K-means : *K* est déterminé à 12
- Les autres paramètres ($\alpha = 1, \beta = 0.8, \tau_0 = 0.001$) : Valeurs empiriques. Sont des valeurs qui ont été définies par des expériences des chercheurs du domaine **[Trejos, 2004] [Trejos, 2007]**.

Ces valeurs sont appliquées pour la classification de l'image étudiée, tableau de données, qui est constituée de *N*=700*800 pixels=560000 pixels.

Tableau 5.18 Résultats d'*ACOClust*

Paramètres	**Valeur exp.1**	**Valeur exp.2**	**Valeur exp.3**	**Résultats**
M	200	300	280	
β	0.8	0.8	0.8	
α	1	1	1	
ρ	0.3	0.3	0.3	
T_{max}	5.10E+5	4.5.10E+5	5.5.10E+5	
τ_0	0.001	0.001	0.001	
K (K-means)	12	12	12	
	12	10	11	Nbre de classes obtenues
	3.08	4.10	3.12	Meilleure $\zeta_{\text{int } ra}$

Ces résultats montrent que malgré l'augmentation du nombre de fourmis et le nombre d'itération, le meilleur résultat est donné par la première expérience qui initialise pour ces trois paramètres des valeurs minimales et résulte à des résultats performants. Ce qui nous pousse à dire que *ACOClust* donne des meilleurs résultats sans augmentation des valeurs des paramètres et que leurs améliorations donnent des résultats contraires. Ce qui est un point fort pour *ACOClust* dans le cas de la complexité algorithmique pour ne pas tenter de changer les paramètres à réduire le temps d'exécution

5.3.6 Les résultats globaux

Plusieurs expériences ont étaient effectuées pour permettre d'appliquer d'un coté les algorithmes références (*AntClass et k-means*) et de l'autre coté de les critiquer et de proposer des adaptations visant à améliorer les insuffisances des anciennes versions.

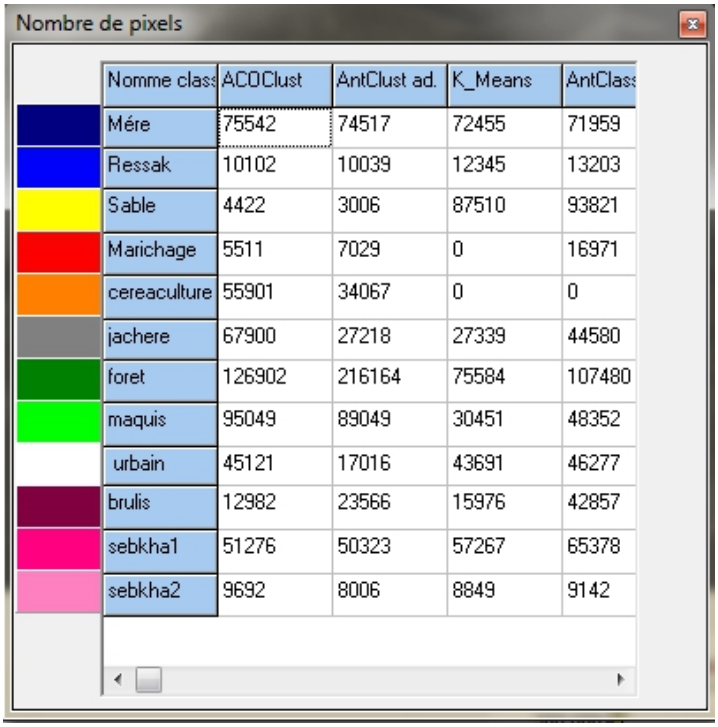
Sur la figure Fig. 5.18.c, nous donnons le résultat de la classification de l'image simulée par l'algorithme classique des *K-means* en initialisant un nombre de classes égale 10.

Ces résultats sont comparés par rapport à la partition initiale (Fig. 5.18)

Les résultats de la comparaison entre *ACOClust, AntClust adapté, AntClass et K-means* ont permis de montrer que les résultats d'*ACOClus*t sont meilleurs des trois autres algorithmes.

Le tableau 5.19 représente le nombre de pixels pour chacune des classes obtenues par les quatre algorithmes.

Tableau 5.19 Nombre de pixels des classes pour chaque algorithme



Nomme class	ACOClust	AntClust ad.	K_Means	AntClass
Mére	75542	74517	72455	71959
Ressak	10102	10039	12345	13203
Sable	4422	3006	87510	93821
Marichage	5511	7029	0	16971
cereaculture	55901	34067	0	0
jachere	67900	27218	27339	44580
foret	126902	216164	75584	107480
maquis	95049	89049	30451	48352
urbain	45121	17016	43691	46277
brulis	12982	23566	15976	42857
sebkha1	51276	50323	57267	65378
sebkha2	9692	8006	8849	9142

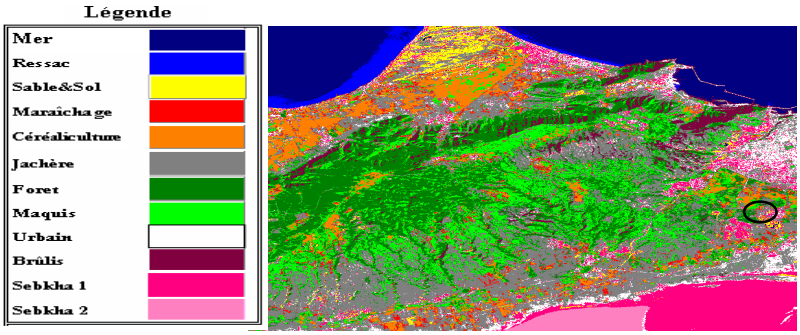
Après calcul du nombre de pixels répartis, on obtient le tableau suivant (tableau 5.23):

Tableau 5.20 Répartition des pixels des méthodes de classification

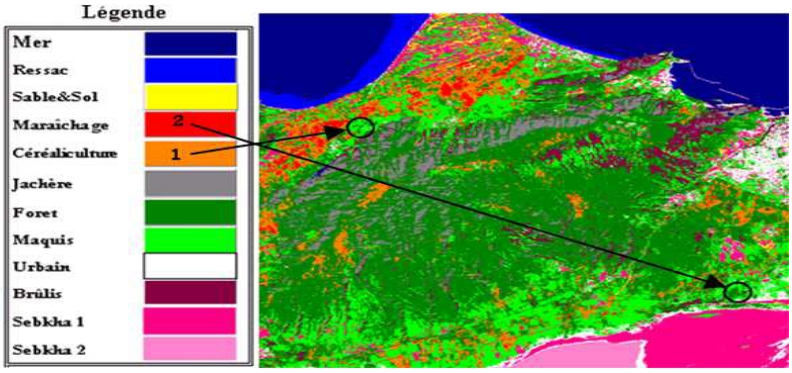
	ACOClust	***AntClust ad.***	***AntClass***	***K-means***
Nb pixel	560000	560000	560000	431437
% expl.	100%	100%	100%	72%

On constate que les algorithmes ***ACOClust***, ***AntClust adapté*** et ***AntClass*** ont exploités tous les pixels de l'image avec existence de 28 % de pixels non exploités pour ***K-means***. Ce qui nous pousse à dire que l'application de l'ACO, dans

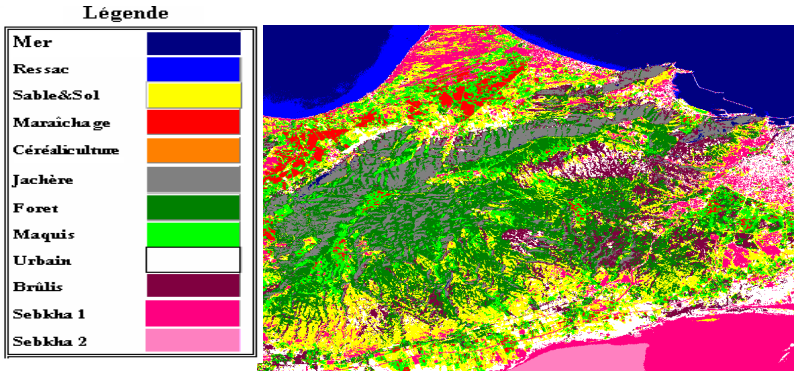
ACOClust, a améliorer k-means. Ce dernier taux a été obtenu quand la partition a convergé vers un minimum local(égalité d'une partition avec sa précédente ou stabilité des centroïdes), sans prise en charge du nombre d'itération. Donc ***ACOClust*** et ***AntClust*** adapté ont obtenus un nombre de classes égal au nombre de niveaux de gris présents dans l'image à classifier (voir Fig. 5.18). Pour la classe maraîchage (2), sa présence dans les Figures Fig. 5.18.a, Fig. 5.18.c et Fig. 5.18.d. Pour la classe Céréaculture (1), sa présence seulement sur la figure 4.11.a et 4.11.d. Ce qui confirme que les algorithmes ***AntClust adapté*** et ***ACOClust*** ont pu extraire le nombre exacte de classes. Les résultats montrent clairement que ces deux algorithmes ***Antclust adapté*** et ***ACOClust*** bien qu'ils ne nécessitent pas une connaissance du nombre probable de classe, arrivent à identifier le nombre égal de classes.



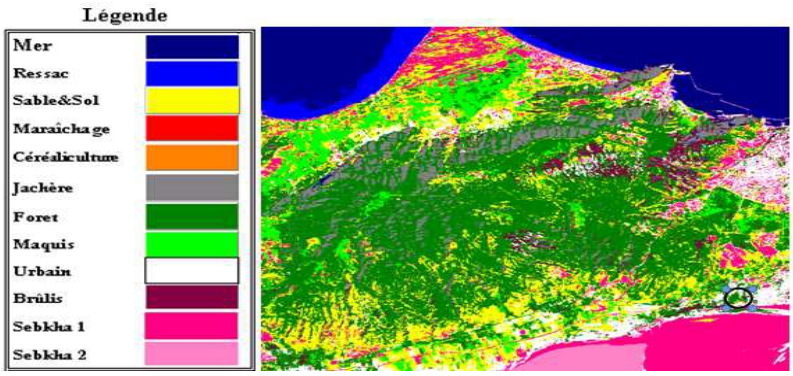
(a)



(b)



(c)



(d)

Fig. 5.18 Image classifiée AntClust adapté (a), par K-Means (b), par AntClass (c) et par ACOClust (d)

Bien que, ***AntClass*** a donné des résultats proches de ***AntClust adapté*** et ***ACOClust*** et que son apport dans ce cas est sans intérêt. Ce qui montre clairement que les algorithmes ***Antclust adapté*** et ***ACOClust*** donnent des résultats performants. Du premier coté l'adaptation d'***AntClust*** et de l'autre coté l'utilisation d'***ACOClust*** ont montrés leur efficacité de classification des images satellites.

5.3.6.1 Evaluation des résultats

Dans le but d'évaluer les classes obtenues par ***ANTClust*** adapté et ***ACOClust***, nous avons utilisé une base de données supervisée dont on connaît le nombre de classes qui sera utilisé pour évaluer les résultats. Nous avons utilisé deux techniques. Nous avons défini trois mesures pour effectuer l'évaluation des résultats (Voir Annexe A). Les deux premières mesures sont :

1. Erreur de classification (E_c) :

$$E_c = \frac{Nombre\ de\ pixels\ mal\ classés}{Nombre\ total\ de\ pixels} \qquad \text{Eq. 5.3}$$

2. Taux de classification (T_c):

$$T_c = (1 - E_c) * 100\% \qquad \text{Eq. 5.4}$$

$$T_c = (\frac{Nombre\ de\ pixels\ bien\ classés}{Nombre\ total\ de\ pixels} * 100)\%$$

Tandis que la troisième mesure est l'index Rand R (Voir Annexe A).

L'index de Rand *R* (Voir Annexe A) est le plus généralement utilisé dans la littérature, dont l'utilisation est répondue pour valider la performance des algorithmes de classification [Ouadfel, 2006].

L'index de Rand prend ses valeurs dans l'intervalle [0,1]. La valeur élevée de *R* (tend vers 1) indique une parfaite correspondance entre la classification de référence et celle obtenue par un algorithme de classification.

Tableau 5.21 Résultats de classification des images satellites par les algorithmes ***ACOClust, AntClust adapté, AntClass et K-means.***

	ACOClust	**AntClust ad.**	**AntClass**	**K-means**
Erreur Ec	0.00711	0.00828	0.0691	0.0817
Taux Tc	99.289 %	99.172 %	93.089 %	91.83 %
Index de R	98.71 %	98.65 %	91.69 %	81.56 %

Les résultats de la comparaison entre ACOClust, AntClust adapté, AntClass et K-means ont permis de montrer que les résultats d'ACOClust sont meilleurs des trois autres algorithmes.

Donc les preuves de l'évaluation par la mesure d'index Rand *R* (Rand *R*= 0,8156) est proche que celle du taux de classification c'est-à-dire le résultat obtenue par l'algorithme ***AntClass et k-means*** ne sont pas performants et l'index Rand *R* justifie ce qui a été dit auparavant que les deux algorithmes ***AntClust adapté et ACOClust*** sont des algorithmes efficaces pour la classification des images satellites.

5.3.7 Classification des données de Machine Learning Repository (voir annexe B2)

5.3.7.1 Introduction :

Après avoir appliqué les algorithmes de colonies de fourmis avec manipulation des images satellites comme données de traitement, on a remarqué que l'algorithme ACOClust est performant que l'algorithme AntClust. A la fin, nous avons soulevé plusieurs points :

- La taille volumineuse des images satellites a posé un problème énorme durant nos expériences. Le temps d'exécution est un exemple marquant de ces problèmes.
- L'indisponibilité de plusieurs données satellites que celle que nous avons reçue du laboratoire dont j'effectue mes recherches. Donc on a seulement extrait des résultats et on n'a pas effectué des comparaisons avec d'autres images satellites.

Suivant ce qui a été vu comme résultats cités ci-dessus, on peut dire qu'on n'a pas effectué plusieurs expériences qui vont nous poussées à dire d'une façon définitive que telle ou telle algorithme est meilleur que l'autre ou non. Pour atteindre l'affirmation finale à cette dernière proposition on été obligé à refaire plusieurs expériences avec d'autre types de données. La Machine Learning Repository est une base très riche de différents types de données qui est offerte aux chercheurs pour leurs propres expériences afin de valider leurs expériences.

Cette deuxièmes parties sera considérée comme un deuxième essai des algorithmes de colonies de fourmis afin d'apprécier les résultats et de conclure par suite sur les caractéristiques des algorithmes. Les résultats cités ci-dessous sont sous forme de tableaux de chaque type de données. Ces tableaux contiennent tous les résultats attendus afin de passer directement à une comparaison approfondis des algorithmes retenus : ACOClust et AntClust.

On va choisir l'ensemble des données **Iris, wine et les lentilles cornéennes** pour tester les algorithmes de classification Antclust et AcoClust.

5.3.7.2 La nature des données et les paramètres utilisés :

Les données utilisées dans cette partie sont de la Machine Learning Repository (Voir annexe B2). Les données choisies sont **Iris, wine et les lentilles cornéennes** pour tester les algorithmes de classification Antclust et AcoClust.

Les algorithmes dépendent d'un certain nombre de paramètres. Tous ces paramètres ont été choisis d'une façon empirique. Le tableau suivant résume les valeurs des paramètres d'AntClust et AcoClust appliquées pour la classification des données étudiée.

Tableau 5.22 Paramètres des algorithmes Antclust et Acoclust

Paramètres	**Description**
AntClust	
Nombre de fourmis	Nombre de fourmis
*k*d,*k*p	contrôle la probabilité de déposer, déplacer un pixel d'une case (*k*d et *k*p sont des constantes positives. Quand $f << kp$ cela signifie qu'il y a peu d'objets dans le voisinage de l'objet et donc la probabilité de le prendre est élevée (*Pp* est proche de 1). Inversement quand $f >> kp$ la probabilité de prendre l'objet est faible s'il est entouré de plusieurs objets.)
P_p	
p_d	Une fourmi ne pourra déposer un objet sur un tas que si l'objet qu'elle transporte est suffisamment similaire au centre de gravité du tas par rapport à un seuil fixé P_d.
B	β est un paramètre qui contrôle la dilatation de la fonction de similarité *f*.
T_{max}	Nombre des itérations de l'algorithme AntClust
AcoClust	
Nombre de fourmis	Nombre de fourmis
T_{max}	Nombre des itérations de l'algorithme AntClust
A	les puissances α de la pheromone
B	les puissances β de l'heuristique locale,
t0	la valeur initiale de phéromone
P	le coefficient d'évaporation
Q	Contrôle la quantité de phéromones déposées par les fourmis

5.3.7.3 Résultats et discussion:

Notre travail expérimental a pour but de comparer les deux algorithmes AntClust et AcoClust, ses algorithmes à base des fourmis se basent sur deux techniques totalement déférentes.

Les résultats ci-dessous ont été obtenus suivant le fonctionnement de l'application, par :

- ➔ L'application d'AcoClust avec un facteur de transferts α = 1, β = 0.8, ρ = 0.3, τ_0= 0.001, Q= 0.5,
- ➔ L'application d'AntClust avec k_d=0.3,k_p=0.1 P_p=0.015, p_d=0.56, B=50,

Afin de tester les performances des algorithmes AntClust et ACOClust nous avons utilisé des données de Machine Learning Repository.

L'intérêt de ces données est qu'on possède d'une manière précise le nombre exact de classes ainsi que la qualité de la classification de référence. Ces informations ne sont évidemment pas transmises aux algorithmes AntClust et ACOClust et ne sont utilisées que pour tester leur capacités à retrouver la partition théorique des objets. D'un autre côté, on peut comparer la qualité de la partition obtenue avec les deux algorithmes et celle de la réalité.

Les applications de AntClust et ACOClust pour la classification des données de ces bases de la Machine Learning Repository conduisent à une évaluation des résultats obtenus en l'absence d'informations sur la partition correcte. Cependant, elles permettent l'analyse de la structure des classes générées en utilisant des mesures de validation.

Les tableaux cités ci-dessous (Tableau 5.23, Tableau 5 .24 et Tableau 5.25) présentent plusieurs points importants. On a choisi les critères suivants utilisés dans les différentes expériences :

- Le nombre maximal d'itération : Tmax,
- Le nombre de fourmi NB_F,
- Le nombre d'objets pour chaque classe C1, C2 et C3 (NB. : on remarque que les trois bases choisis ont un nombre identique de classes : 3),
- Le temps d'exécution T. Afin de prendre une idée sur le temps effectué pour obtenir les partitions je suis obligé de citer les caractéristiques de la machine que j'ai utilisée car le temps d'exécution peut changer d'une machine à une autre et ceci suivant leurs caractéristiques internes. (***n*** hs : signifie n/100 seconde. ex. 3 hs=3/100 s)
- Inertie : c'est l'inertie intraclasse. L'inertie intraclasse mesure la séparation entre les classes d'une partition. Plus l'inertie intraclasse est petite plus les classes sont séparées. Ce paramètre est un outil d'arrêt de l'algorithme et ceci lorsqu'on obtient, après un nombre d'itération, une valeur maximale et on obtient la meilleure partition.
- RandR : Index de Rand R. c'est le taux d'évaluation choisi.

Pour évaluer les résultats de classification obtenus par les deux algorithmes AntClust et ACOClust, nous avons choisi la mesure l'index de Rand dont l'utilisation est répondue pour valider la performance des algorithmes de classification. Ainsi, pour étudier la performance de ces deux algorithmes, nous avons décidé d'utiliser la mesure d'inter classes. En plus de ces deux mesures, pour chaque expérimentation, nous donnons le temps d'exécution effectué pour chaque algorithme et le nombre d'objets de chaque classe identifiée.

En considérant les données iris, wine et les lentilles cornéennes nous avons mené une série de tests en faisant varier les deux paramètres : le nombre d'itérations maximales et le nombre des fourmis. (On s'est basé sur ces deux paramètres parce que les autres n'influent pas clairement sur les résultats de classification). Tous les résultats des deux algorithmes AntClust et ACOClust ont été rassemblés sur les tableaux cités dessous.

Tableau 5 .23. Résultats de la classification des données iris obtenues par les algorithmes AntClust et AcoClust

Méthodes / Paramètres		AntClust						AcoClust					
Tmax	NB_F	C1	C2	C3	T	inertie	Rand R	C1	C2	C3	T	Inertie	Rand R
5	**3**	136	13	1	5 hs	4,55	0,50	*39*	*61*	*50*	*15 hs*	*1,58*	*0,91*
5	**12**	144	5	1	6 hs	3,06	0 ,55	*39*	*61*	*50*	*49 hs*	*4,23*	*0,91*
5	**24**	117	30	3	7 hs	4,76	0,62	*32*	*68*	*50*	*78 hs*	*3,21*	*0,87*
20	**3**	*107*	*41*	*2*	*7 hs*	*4,55*	*0,51*	132	13	5	43 hs	1,58	0,50
20	**12**	145	2	3	8 hs	3,97	*0,54*	132	13	5	1 s 51 hs	4,23	0,50
20	**24**	112	14	24	9 hs	7,89	*0,52*	109	35	6	2 s 9 hs	3,21	0,54
30	**3**	*63*	*78*	*9*	*9 hs*	2,71	*0,79*	112	32	6	64 s	1,58	0,47
30	**12**	*67*	*67*	*16*	*10 hs*	4,10	*0,88*	121	27	2	2 s 20 hs	4,12	0,43
30	**24**	*4*	*80*	*66*	*11 hs*	8,76	*0,84*	130	17	3	4 s 32 hs	3,21	0,41

Tableau 5.24. Résultats de la classification des données Wine obtenues par les algorithmes AntClust et AcoClust

Méthodes / Paramètres		AntClust						AcoClust					
Tmax	NB_F	C1	C2	C3	T	inertie	Rand R	C1	C2	C3	T	inertie	Rand R
5	**3**	18	135	25	1 hs	2,42	0,44	49	102	27	1 hs	3	0,54
5	**12**	21	130	27	3 hs	2,89	0,42	*61*	*69*	*48*	*5 hs*	*2, 36*	0,74
5	**24**	16	136	26	-	-	-	31	122	25	6 hs	2,04	0,44
20	**3**	22	132	24	5 hs	2,72	0,45	38	110	30	7 hs	3	0,34
20	**12**	21	134	23	-	-	-	*61*	*69*	*48*	*9 hs*	*1,57*	0,96
20	**24**	15	149	34	8 hs	6,2	0,42	48	105	25	10 hs	1,63	0,51
30	**3**	7	40	131	7 hs	3,20	0,34	15	140	27	9 hs	2,8	0,51
30	**12**	52	81	45	-	-	-	38	115	25	14 hs	2,3	0,51
30	**24**	18	3	157	-	-	-	99	52	27	16hs	1,43	0,36

Tableau 5.25. Résultats de la classification des données – les lentilles cornéennes- obtenues par les algorithmes AntClust et AcoClust

Méthodes / Paramètres		AntClust						ACoClust					
Tmax	NB_F	C1	C2	C3	T	inertie	Rand R	C1	C2	C3	T	inertie	Rand R
5	**3**	9	10	5	*14hs*	49927,5	0,65	*6*	*6*	*12*	*28hs*	48052,4	0,59
5	**12**	18	1	5	15*hs*	49666,6	0,45	*12*	*6*	*6*	*76hs*	47019	0,74
5	**24**	3	19	2	17*hs*	95059,5	0,55	*18*	*3*	*3*	*1s 36hs*	44345	0,87
20	**3**	17	4	3	16*hs*	82758,4	0,50	*6*	*6*	*12*	*77hs*	48052,4	0,59
20	**12**	18	4	2	17*hs*	132317	0,53	19	1	4	2s 58*hs*	47019	0,71
20	**24**	6	12	6	18*hs*	161130	0,65	5	5	14	2s 12*hs*	42345	**0,96**
22	**3**	10	6	8	19*hs*	**79697,7**	0,75	*12*	*6*	*6*	*1s 5hs*	**48052,4**	0,45
22	**12**	6	7	11	21*hs*	**143635**	0,83	***20***	***2***	***2***	***3s 39hs***	**44019**	0,68
22	**24**	8	6	10	22*hs*	**312535**	0,89	20	1	3	7s 25*hs*	**39345**	0,80

Remarque : le symbole '-' signifie que l'application ne se termine pas (se bloque) si le nombre de fourmis s'agrandit (supérieur à 3).

A travers une lecture des tableaux précédents on remarque les points suivants :

- Si le nombre d'itération et de fourmis augmente le temps d'exécution augmente,
- Si le nombre d'objets se répartissent suivant le nombre réel le taux Rand R s'améliore,
- Le taux Rand R s'améliore si l'inertie diminue.
- Les valeurs du taux Rand R de l'ACOCust suivant les mêmes valeurs de AntClust donnent des meilleurs résultats (0,96).

On conclu qu'à travers ces résultats qu'AcoClust trouve de bons résultats avec un nombre très petit de fourmis (<10). On remarque que le nombre d'itération n'influe sur les résultats. On constate l'incapacité d'AntClust, à extraire la bonne classification, il trouve des résultats approchés surtout quand la nature des données est totalement entier (***MLR*** : les lentilles coréennes), mais reste insuffisants même que le temps d'exécution est presque négligeable.

5.3.8 Conclusion

AntClust et ACOClust sont des nouvelles approches pour la classification non supervisée de différents types de données. Elles sont inspirées du comportement de tri de couvain observé chez les fourmis réelles. L'intérêt principal de ces algorithmes est leur capacités à extraire automatiquement les classes présentes dans l'ensemble de données sans connaître le nombre de classe a priori et sans partition de départ. Dans *AntClust adapté,* l'adaptation était de permettre d'atteindre cet objectif. Dans ***ACOClust***, on a pu savoir qu'il est efficace à classifier des différents types de données.

Les expériences effectuées montrent que ces deux algorithmes, ***AntClust adapté et ACOClust***, fournissent de très bons résultats et s'avèrent comme prévu supérieur aux algorithmes ***K-means et AntClass***.

En conséquence, le tri collectif du couvain n'est pas la seule source d'inspiration que l'on peut obtenir des fourmis pour la classification mais une autre source qui est la reconnaissance chimique. Le mécanisme de reconnaissance inter-individuels et la constitution de l'odeur coloniale (la phéromone) peuvent aussi s'apparenter à des mécanismes de classification.

Différencier ses congénères de ses ennemis tout en modifiant constamment son propre schéma d'identification peut s'apparenter à la détermination des objets qui font partie d'une même classe d'autres objets.

Finalement, l'évaluation des résultats obtenus par ***AntClust et ACOClust***, nous permet de dire qu'après une comparaison avec les autres méthodes (***AntClass et K-means***) qu'ils sont plus efficaces.

5.4 Conclusion générale du chapitre

Dans ce chapitre, nous avons tenté de prouver, à travers l'application des algorithmes de colonies de fourmis suivant des exemples variés, que le paradigme des fourmis pouvait être fécond dans divers domaines. En effet, nous avons présenté plusieurs applications divisées en deux grandes parties : optimisation et classification.

Suivant ce qu'on a vu que les fourmis possèdent une multitude d'algorithmes qui ont été utilisés dans plusieurs domaines et problèmes et nous avons senti qu'elles sont capables de trouver des solutions encourageantes par rapport à d'autres techniques (heuristiques ou méta-heuristiques).

Même si les algorithmes de fourmis sont de plus en plus utilisés dans différentes domaines, le potentiel des fourmis artificielles n'est pas encore pleinement exploité par les biologistes. Car leurs inspirations pour la conception de méthodes de résolution de problèmes complexes à une forme de vie artificielle peuvent apporter beaucoup dans l'étude de la vie.

Suivant ce qu'on a vu durant ce chapitre, on a vu des expériences sur des problèmes d'optimisations et de classifications.

Finalement, au regard des résultats obtenus, l'application des algorithmes de colonies de fourmis ont données des résultats performants par rapport à ceux d'autres méthodes.

Conclusion générale et perspectives

Chapitre 6 : Conclusion générale et perspectives

Le thème de notre thèse est l'étude de différents algorithmes de colonies de fourmis conçus spécialement pour l'optimisation et la classification. L'objectif visé, en premier lieu d'appliquer ces algorithmes et après les étudié et de les adapter au problème traité afin de prouver leurs possibilités de résoudre différents types de situations.

Les méta-heuristiques sont souvent inspirées par des systèmes naturels, et les algorithmes de colonie de fourmis forment une nouvelle classe ajoutée aux méta-heuristiques récemment proposée.

Les méthodes intelligentes peuvent apporter un plus à des problèmes d'optimisation d'une part et d'améliorer les performances de la classification d'autre part. Les méthodes traditionnelles ont des limitations dans la construction de méthodes capable d'arriver à des solutions efficaces.

Toutefois, notre étude de ces méta-heuristiques n'avait pas pour but de les présenter comme une alternative immédiate aux approches classiques, car, lorsque le modèle distributionnel des données est connu, l'efficacité de ces approches est garantie. Notre étude vise à présenter les méta-heuristiques comme des méthodes de résolution approchées se voulant simples et adaptables à tout type de problèmes non modélisables. Leur capacité à optimiser un problème avec un minimum d'informations est équilibrée par le fait qu'elles n'offrent aucune garantie quant à l'optimalité de la solution trouvée.

Du point de vue de la recherche opérationnelle, cet inconvénient n'est pas toujours un problème, tout spécialement quand une seule approximation de la solution optimale est recherchée. Nous aurions pu mettre en évidence l'avantage de ces nouvelles approches en utilisant différents types de données. La diversification des données a été dans la mesure où on peut montrer compris le type image qu'on a voulu qu'elle sera l'unique.

En effet, nous avons vu que les fourmis en colonie présentent, des comportements auto-organisés, où des interactions simples au niveau local permettent l'émergence d'un comportement global complexe.

Les premiers travaux dans le domaine de l'utilisation de ces caractéristiques en recherche opérationnelle ont donne lieu a des méta-heuristiques d'optimisation pour des problèmes combinatoires.

Par suite, plusieurs chercheurs ont inspirés plusieurs autres sources d'inspirations des fourmis.

Deneubourg de son tri collectif du couvain qui n'est pas la seule source d'inspiration que l'on peut obtenir des fourmis pour la classification mais il existe une autre source qui est la reconnaissance chimique. La recherche de nourriture par exemple, peut être considérée comme une compétition entre plusieurs colonies pour le contrôle de certaines zones de chasse.

Le mécanisme de reconnaissance inter-individuels et la constitution de l'odeur coloniale (la phéromone).

On pourrait décrire cette méta-heuristique comme un système distribué où les interactions entre composants, par le biais de processus stigmérgiques, permettant de faire émerger un comportement global cohérant rendant le système capable de résoudre des problèmes d'optimisation difficile.

Les algorithmes des colonies de fourmis possèdent des caractéristiques intéressantes telles que le parallélisme intrinsèque élevé, la flexibilité (la colonie s'adapte à des modification de l'environnement), la robustesse (une colonie est apte à maintenir son activité si quelques individus sont défaillants) en plus de la décentralisation et de l'auto organisation, ce qui rend cette démarche plus utile pour les problèmes difficiles tels que la classification non supervisée et la recherche de l'optimum global dans les problèmes d'optimisation combinatoire.

Notre contribution dans cette recherche se situe à plusieurs niveaux, nous citons, en particulier:

- Possibilité d'adapter des algorithmes et de permettre d'obtenir de meilleurs résultats ;

- La proposition d'une méthode pour la l'obtention d'une partition de bonne qualité;

- Application des algorithmes de colonies de fourmis pour la classification qui a été exploité pour deux types de données : les images satellites et la trois modèles de Machine Learning Repository;

- Application des algorithmes de colonies de fourmis pour l'optimisation qui a été exploité à travers trois visions : la première traite le problème TSP "Att48", qui représente une carte des grandes villes des États-Unis avec l'algorithme AS, la seconde étude le même problème TSP mais sur plusieurs types de données avec ACO et finalement l'application de la méthode de colonies de fourmis ACO pour l'optimisation d'un problème réel : réseau électrique;

Nos perspectives de recherche sont mentionnées comme suit :

- Réglage de paramètre automatiquement des paramètres des méta-heuristiques étudiées et aussi pour maîtriser leurs fonctionnements.

- Exploiter le parallélisme implicite des algorithmes basés fourmis afin d'améliorer les temps de convergence. Par exemple, on pourra associer aux fourmis des processeurs indépendants qui communiqueront entre eux à travers une mémoire commune qui contiendra les valeurs des phéromones.

Annexes

Annexe A :Méthodes d'évaluation des résultats de classification

A 1.Critères d'évaluation des méthodologies proposées [Ouad, 2006]:

Les différentes méthodologies non supervisées proposées dans cette recherche aboutissent à une carte d'occupation du sol. L'évaluation des résultats de classification obtenus dans notre travail peut être effectuée par deux approches différentes. Dans la **première approche**, un résultat de classification est évalué en prenant en compte des informations a priori (supervisée). L'évaluation consiste à mesurer l'écart entre le résultat fourni par l'algorithme de classification et une classification de référence fourni par une classification manuel. Ceci revient à effectuer une mesure de dissimilarité entre deux classifications. **La deuxième approche** consiste à se référer à l'objet original en évitant de se rapporter à un résultat de référence. Dans le cas de la classification ceci revient à se fixer un modèle de l'erreur de mesure et à estimer la qualité du résultat par rapport à ce modèle.

Ces deux mesures d'évaluation utilisées dans nos méthodes de classification

A 2. Evaluation par comparaison avec une classification de référence [Ouadfel, 2006]

La performance des algorithmes de classification est évaluée en comparant le résultat fourni par l'algorithme avec une classification de référence.

La comparaison entre deux résultats de classification revient à effectuer des mesures de dissimilarité entre ces deux classifications. Ces mesures prennent en compte dans leur formulation une information sur la taille des classes mal classées et sur leur taux de recouvrement ainsi que l'information spatiale concernant la localisation des objets qui la constituent. Une grande valeur de ces mesures indique une grande erreur de l'algorithme de classification et donc une performance faible de l'algorithme.

Voici la liste des notations utilisées :

Soit $\boldsymbol{I}$ l'ensemble des données de $\boldsymbol{N}$ objets. Soit $\boldsymbol{I_{part}}$ la classification de l'ensemble des données $\boldsymbol{I}$ est partitionnée en $\boldsymbol{C_{part}}$ classes et $\boldsymbol{I_{Ref}}$ la classification de référence en $\boldsymbol{C_{Ref}}$ classes que l'on souhaite obtenir. On note $\boldsymbol{C_{part}\,(i)}$ et $\boldsymbol{C_{ref}\,(i)}$ l'étiquette de la classe de l'objet o_i respectivement dans $\boldsymbol{I_{part}}$ et $\boldsymbol{I_{Ref}}$.

Notre critère d'évaluation déterminent le taux d'objets bien classés en calculant la différence entre la classification de référence et celle obtenue par un algorithme de classification. Pour cela elles comparent toutes les paires d'objets et vérifient à chaque fois si ces objets sont classés simultanément dans les deux classifications. L'index de Rand R est le plus généralement utilisé dans la littérature et est défini comme suit :

$$R = \frac{a+d}{a+b+c+d} \qquad \text{Eq.A2.1}$$

avec a, b, c et d des paramètres calculés pour tous les couples d'objets O_i et O_j de la façon suivante :

$$
\begin{aligned}
a &= \left|\left\{i, j / C_{ref}(i) = C_{ref}(j) \wedge C_{part}(i) = C_{part}(j)\right\}\right| \\
b &= \left|\left\{i, j / C_{ref}(i) = C_{ref}(j) \wedge C_{part}(i) \neq C_{part}(j)\right\}\right| \\
c &= \left|\left\{i, j / C_{ref}(i) \neq C_{ref}(j) \wedge C_{part}(i) = C_{part}(j)\right\}\right| \\
d &= \left|\left\{i, j / C_{ref}(i) \neq C_{ref}(j) \wedge C_{part}(i) \neq C_{part}(j)\right\}\right|
\end{aligned}
\qquad \text{Eq.A2.2}
$$

L'index de Rand prend ses valeurs dans l'intervalle [0,1]. La valeur élevée de R (tend vers 1) indique une parfaite correspondance entre la classification de référence et celle obtenue par un algorithme de classification.

Différentes variations de l'index de R existent dans la littérature. Parmi elles on cite le coefficient de Jaccard qui applique une stricte correspondance pour chaque classe entre les deux classifications et est définit comme suit :

$$
J^k(I_{part}, I_{ref}) = \frac{\left|V^k_{I_{part}} \cap V^k_{I_{ref}}\right|}{\left|V^k_{I_{part}} \cup V^k_{I_{ref}}\right|} \qquad \text{Eq.A2.3}
$$

Où Soit $V^k_{I_{part}}$ et $V^k_{I_{ref}}$ qui représentent respectivement le nombre total d'objets appartenant à la classe C_k dans les deux partitionnements I_{part} et I_{ref} . Une bonne classification est obtenue quand $J^k(I_{part}, I_{ref})$ est proche de 1 ce qui signifie que la classe C_k a été bien extraite.

Ce type de mesure présente l'inconvénient de ne compter que les objets, sans prendre en considération les propriétés spatiales inhérentes aux objets mal classés.

A 3. L'évaluation se référant aux données originales [Ouadfel, 2006]

Les méthodes présentées dans cette section permettent une évaluation non supervisée des résultats de classification, dans le cas où aucune vérité terrain ne serait disponible. Ces méthodes tiennent compte des points suivants: l'homogénéité intra-classe, l'homogénéité interclasse.

A 3.1. Critère de qualité et notion d'inertie

Une partition pour être bonne doit satisfaire les deux critères suivants :

1. Les individus proches doivent être regroupés.
2. Les individus éloignés doivent être séparé.

Pour évaluer l'homogénéité d'une classe de points où la distance entre deux points est la distance euclidienne, une mesure classique est l'inertie de cette classe pour la distance euclidienne.

A 3.2. Inertie interclasse et inertie intraclasse

On appelle inertie totale d'un classe $\Gamma = \{M_i, i=1\ldots,n\}$ la moyenne des carrés des distances de ses points au centre de gravité du classe. Donc. Si G désigne le centre de gravité de Γ, l'inertie totale de Γ est :

$$
\zeta(\Gamma) = \sum_{i=1}^{n} p_i d(M_i, G)^2 \qquad \text{Eq.A2.4}
$$

Si tous les points de la classe sont de même poids égal à1/n

$$\zeta(\Gamma)=\frac{1}{n}\left(d(M_1,G)^2+d(M_2,G)^2+...+d(M_n,G)^2\right) \qquad \text{Eq.A2.5}$$

L'inertie totale d'une classe est :

$$\zeta(\gamma_j)=\sum_{i\left|M_i\in\gamma_j\right.}^{n} p_i d(M_i,G)^2 \qquad \text{Eq.A2.6}$$

L'inertie est une mesure de l'homogénéité d'un ensemble de points (classe).Une classe sera d'autant plus homogène que son inertie totale sera faible.

Ce qui nous intéresse n'est pas l'homogénéité d'une classe mais l'homogénéité de l'ensemble des classes d'une partition. Pour cela, on introduit la notion d'inertie intraclasse d'une partition :

$$\zeta_{intra}(\Gamma)=\sum_{j=1}^{k}\zeta(\gamma_j) \qquad \text{Eq.A2.7}$$

L'inertie intraclasse d'une partition est la somme des inerties totales de chaque classe de la partition.

L'inertie intraclasse mesure l'homogénéité de l'ensemble des classes. Plus l'inertie intraclasse est faible, plus la partition est composée de classes homogènes. On peut noter deux cas particuliers :

Si chaque classe de la partition représente un seul individu

$$\Gamma=(\gamma_1=M_1,\gamma_2=M_2,...,\gamma_n=M_n)$$

L'inertie intraclasse de cette partition est nulle :

$$\zeta_{intra}(\Gamma)=0 \qquad \text{Eq.A2.8}$$

En effet, comme le centre de gravité d'une classe composée d'un seul point est le point lui même, L'inertie totale de chaque classe est nulle :

$$\zeta(\gamma_j)=d_2(M_i,M_j)^2=0$$

L'inertie intraclasse de la partition est égale à l'inertie totale de la classe :

$$\zeta_{intra}(\Gamma)=\zeta(\Gamma)$$

On introduit maintenant une mesure du niveau de séparation entre les classes :
L'inertie interclasse.

On suppose qu'une bonne mesure du niveau de séparation des classes est la somme pondérée des distances entre de gravité de chaque classe et le centre de gravité de la classe :

$$\zeta_{inter}(\Gamma) = \sum_{j=1}^{k} \mu_j d(G_j, G)^2 \qquad \text{Eq.A2.9}$$

L'inertie interclasse mesure la séparation entre les classes d'une partition. Plus L'inertie interclasse est grande plus les classe sont séparées. On peut noter deux cas particuliers :

Si chaque classe de la partition représente un seul individu

$$\Gamma = (\gamma_1 = M_1, \gamma_2 = M_2, ..., \gamma_n = M_n)$$

L'inertie interclasse de cette partition est L'inertie totale de Γ

$$\zeta_{inter}(\Gamma) = \zeta(\Gamma)$$

Si la partition n'est composée que d'une seule classe

$$\Gamma = (\gamma_1) \text{ où } \gamma_1 = (M_1, M_2, ..., M_n)$$

L'inertie interclasse de la partition est égale à l'inertie totale de la classe :

$$\zeta_{inter}(\Gamma) = \zeta(\Gamma)$$

Notre critère de qualité pour assurer une partition de classe homogènes et distinctes est alors de minimiser l'inertie intraclasse des partitions et de maximiser l'inertie interclasse.

Le théorème suivant nous donne une relation entre inertie intraclasse et inertie interclasse.

- **Décomposition de Huygens**

$$\zeta(\Gamma) = \zeta_{intra}(\Gamma) + \zeta_{inter}(\Gamma) \qquad \text{Eq.A2.10}$$

Pour chaque partition, l'inertie intraclasse plus l'inertie interclasse est égale à l'inertie totale de notre classe de points.

Cette relation permet de faire l'interprétation suivante :

Minimiser l'inertie intraclasse est équivalent à maximiser l'inertie interclasse. Notre critère de qualité est alors soit minimiser l'inertie intraclasse, soit maximiser l'inertie interclasse. On choit ici de minimiser l'inertie intraclasse.

Annexe B : Données manipulées dans notre travail

B 1. Types de données manipulés : images satellites

B.1.1 Introduction

L'étape fondamentale dans une méthode de classification est la représentation des données sous un format analysable par les algorithmes de classification.

La classification intervient sur des données qui résultent d'une suite de choix qui vont influencer les résultats de l'analyse. Classiquement, les données sont décrites dans un tableau de données individus-variables.

Le type de donnée manipulée dans mes travaux est l'image satellite.

Une image est un support d'information. Elle présente les éléments d'une scène qui a été captée, soit par un appareil photographique, soit par un **satellite [benya, 2008].**

Les images satellites sont généralement issues de capteurs. **[Fig. B.1]** Mathématiquement parlant, une image est une application F d'un sous ensemble :

($M \times N$) de ($\mathbb{R} \times \mathbb{R}$) vers l'ensemble des réels ($\mathbb{R}$), qui, à chaque couple de réels (i,j) associe le réel $f(i,j)$. $F : (M \times N) \rightarrow \mathbb{R}$

$$(i,j) \rightarrow f(i,j).$$

Le sous ensemble ($M \times N$) est constitué de couples d'entiers (i,j) tels que : $i \in [O,1 \ldots C_p]$ et $j \in [0,1 \ldots L_n]$.

Une image numérique est à la base un tableau à deux dimensions C_p et L_n qui y représentent son nombre de colonnes et de lignes.

La plupart des images utilisées en télédétection ont des niveaux de gris variant entre 0 et 255.

Une image multispectrale est un ensemble de matrices ou canaux (un canal par longueur d'onde) où sont enregistrées les mesures associées à chaque pixel.

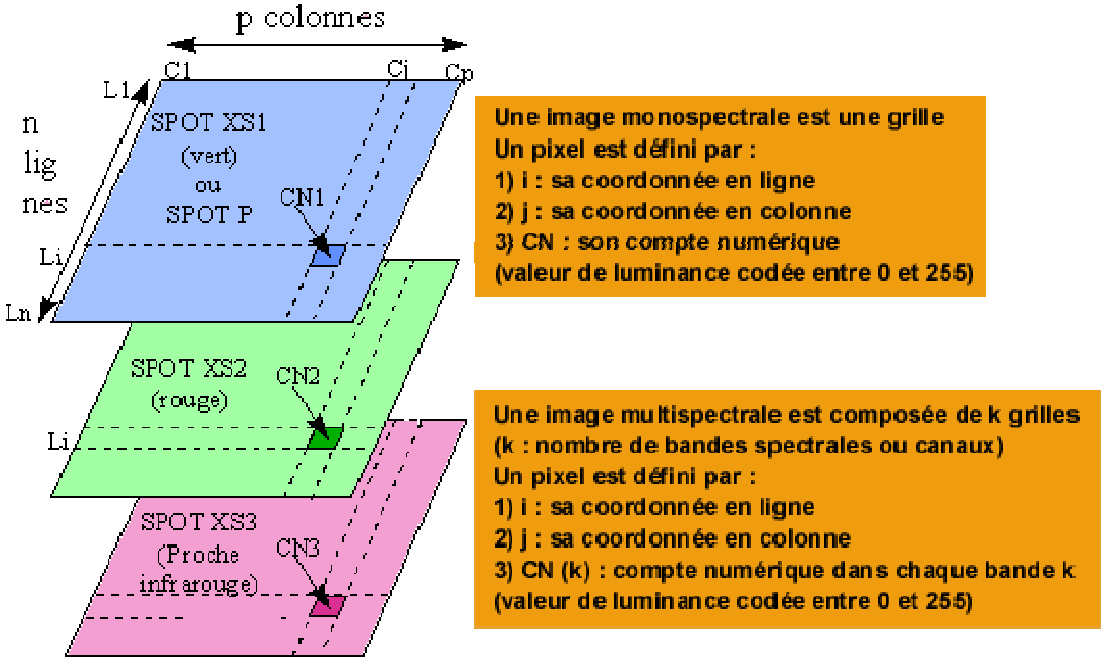


Fig. B.1. Structure d'une image satellite (cas d'une image SPOT XS multispectrales)

B.1.2 Caractéristiques de l'image traitée

L'image Thématic Mapper prise par le satellite LANDSAT 5 le 15 Mars 1993 à 9h45, la région d'Oran ouest (Fig. B.4).

L'image générée par ce satellite est de taille 700 lignes x 800 colonnes de la scène.

Les caractéristiques des différents canaux du satellite LANDSAT 5 sont :

Tableau B.1 Caractéristiques du satellite LANDSAT 5

Type de capteur	Bandes	Résolution spectrale (µm)	Résolution Spatiale (métre)	Applications
TM	TM1 (visible bleu)	0.45-0.52	30 x 30	Discrimination entre le sol et la végétation, bathymétrie / cartographie côtière ; identification des traits culturels et urbains
	TM2 (visible vert)	0.52-0.60	30 x 30	Cartographie de la végétation verte (mesure le sommet de réflectance) ; identification des traits culturels et urbains
	TM3 (visible rouge)	0.63-0.69	30 x 30	Discrimination entre les espèces de plantes à feuilles ou sans feuilles ; (absorption de chlorophylle) ; identification des traits culturels et urbains
	TM4 (proche	0.76-0.90	30 x 30	Identification des types de végétation et de plantes ; santé et contenu de la

	infrarouge)			masse biologique ; délimitation des étendues d'eau ; humidité dans le sol
	TM5 (infrarouge à courtes longueurs d'ondes)	1.55-1.75	30 x 30	Sensible à l'humidité dans le sol et les plantes ; discrimination entre la neige et les nuages
	TM6 (infrarouge thermique)	10.40-12.50	120 x 120	Discrimination du stress de la végétation et de l'humidité dans le sol relié au rayonnement thermique ; cartographie thermique
	TM7 (infrarouge à grandes longueurs d'ondes)	2.08-2.35	30 x 30	Discrimination entre les minéraux et les types de roches ; sensible au taux d'humidité dans la végétation

Tableau B.2 Données statistiques relatives aux canaux TM

Canal	Min	Max	Moyenne	Ecart-Type
TM1	0	255	93.09	61.76
TM2	6	255	107.93	56.32
TM3	4	255	102.55	60.78
TM4	7	255	149.20	71.32
TM5	0	255	112.18	60.73
TM6	30	255	161.19	53.59
TM7	0	255	95.97	57.04

Tableau B.3 Corrélation entre les différents canaux TM

Canal	TM1	TM2	TM3	TM4	TM5	TM6	TM7
TM1	1						
TM2	0.95	1					
TM3	0.71	0.84	1				
TM4	0.86	0.84	0.69	1			
TM5	0.81	0.81	0.71	0.98	1		
TM6	0.81	0.69	0.80	0.70	0.97	1	
TM7	0.74	0.68	0.51	0.92	0.94	0.94	1

On remarque d'un coté, que les canaux TM1 et TM2, TM4 et TM5, TM4 et TM7, TM5 et TM6, TM5 et TM7, TM6 et TM7 sont fortement corrélés puisque les coefficients de corrélation valent respectivement 0.95, 0.98, 0.92, 0.97, 0.94 et 0.94. D'un autre coté, on remarque que les coefficients de corrélation entre les sept (07) canaux TM sont tous élevés (**Voir Tableau B.3)** ce qui signifie qu'ils sont corrélés, c'est-à-dire qu'ils contiennent pratiquement les mêmes informations.

Pour réaliser notre étude, on a besoin d'une image trichromie de la scène LANDSAT. Le choix des trois images qui vont composer cette image nous a mené à faire la correspondance entre la réflectance des objets les plus usuels (eau, végétal, sol) et les longueurs d'ondes les plus usuelles des thèmes d'intérêt (Fig.B.2).

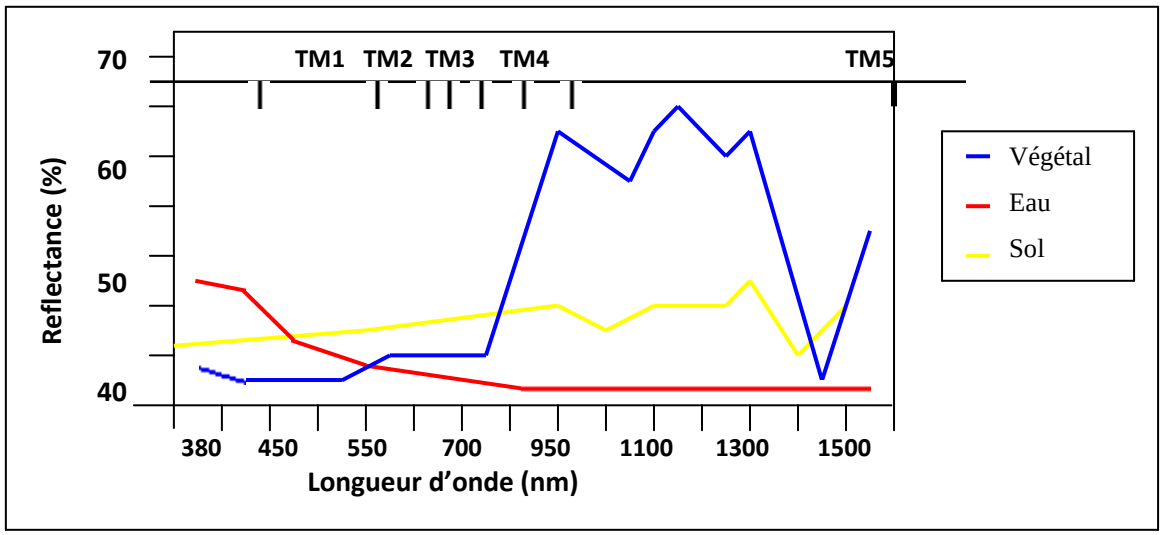
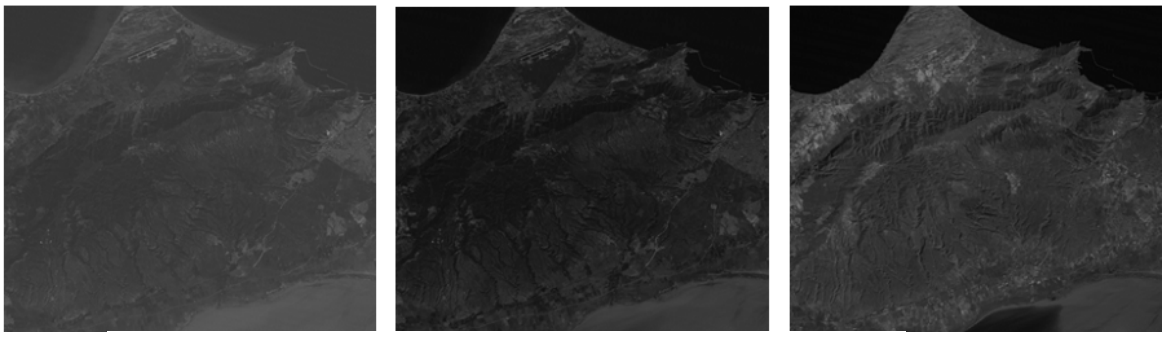


Fig. B.2 Réflectance des objets thématiques

Après analyse, nous avons remarqué que :

- Pour l'eau : les plus fortes valeurs sont dans le canal TM1.
- Pour la végétation : le canal le plus proche d'un pic d'absorption est le canal TM4.
- Pour le sol : la réflectance est forte en canal TM4.
- Le canal TM3 est, parmi tous les canaux qui restent (TM2, TM3, TM5 et TM7), le plus décorrélés avec les autres canaux.

Finalement, la portion d'image étudiée sera une image trichromie (Fig. B.4). Composée des trois canaux TM1, TM3 et TM4 (Fig. B.3).



Canal 1 **Cana 3** **Canal 4**

Fig. B.3 Images des canaux 1, 3 et 4 pour la région d'Oran

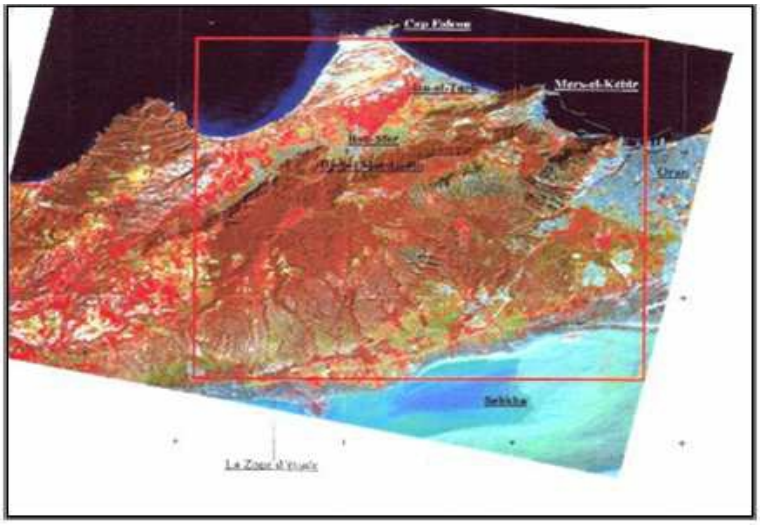


Fig. B.4 Composition colorée des canaux TM 1,3 et 4

La figure suivante présente l'histogramme des images des canaux 1,3 et 4.

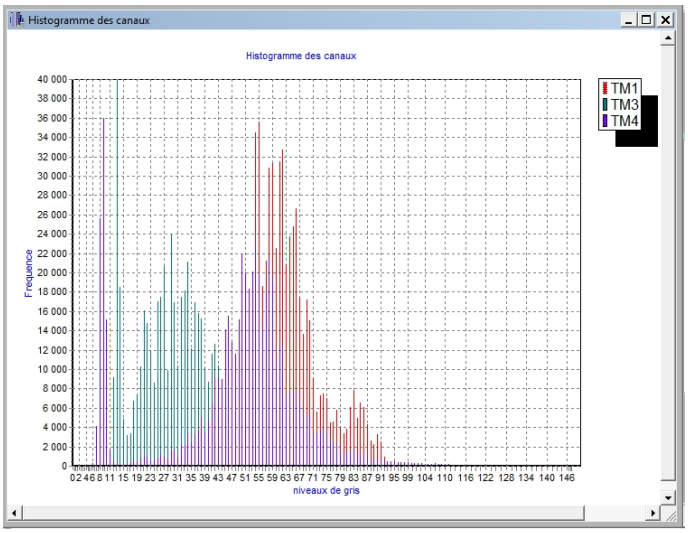


Fig. B.5 Histogramme des images des canaux 1,3 et 4

B 1.3 Données statistiques relatives aux échantillons utilisés :

Les méthodologies de classification qui font objet de notre travail sont non supervisées et ne sont pas basées sur une connaissance a priori sur le site d'étude et les différents thèmes qui s'y trouvent. Autrement dit, ces méthodologies ne requièrent pas une base d'entrainement pour la mise en œuvre de l'algorithme. Cette base de contrôle, pour notre cas, a pour effet que sur l'évaluation statistique du résultat obtenu. L'extraction de ces bases est généralement effectuée par rapport à des données de référence qui peuvent être des cartes thématiques, des photographies aériennes, des images déjà classifiées, des données prises sur le terrain, ou de l'avis d'un expert connaissant le site d'étude « thématicien ».

Les différents thèmes contenus dans l'image trichromie sont au nombre de 11 et sont définis comme suit : Mer, Ressac, Sable, Maraîchage, Céréaliculture, Jachère, Forêt, Maquis, Urbain, Brûlis, Sebkha.

Un jeu d'échantillon, choisis dans l'image initiale à partir des régions identifiées visuellement, a été utilisé, pour la phase d'évaluation.

Le choix de la taille des échantillons n'est pas aléatoire, en effet, pour ce faire nous avons pris en compte deux critères :

Premièrement, si la taille de la classe est relativement petite par rapport à celle des autres classes figurant sur l'image, l'échantillon correspondant à cette classe aura une taille relativement petite par rapport aux autres échantillons.

Ce principe a été appliqué dans le choix de l'échantillon relatif à la classe Ressac (échantillon de taille 23 pixels). Notons que l'inverse n'est pas toujours vrai et que ce cas est soumis à un deuxième critère :

Le deuxième critère repose sur la nature de la classe. Dans le cas où la classe est homogène (Classe Mer par exemple), un échantillon de taille petite peut assurer une représentation fidèle de la classe en question quelque soit sa taille. Mais dans le cas où la classe est hétéroclite, un échantillon composé de plusieurs portions distribuées sur toute la région est nécessaire pour assurer la bonne représentation de la classe (c'est le cas des classes : Urbain et Sebkha).

Les statistiques relatives à ces échantillons sont mentionnées dans le tableau B.4 ci-dessous :

Tableau B.4 Données statistiques à ces échantillons

Thèmes	Bandes	Min	Max	Moyenne	Ecart-Type	Nombre de pixel
Mer	TM4	8	10	9.90	0.40	76
	TM3	12	15	13.50	0.62	
	TM1	52	59	55.68	1.39	
Maquis non exposé	TM4	34	72	46.88	9.17	96
	TM3	17	51	21.50	4.68	
	TM1	51	77	55.53	4.00	
Urbain	TM4	17	74	48.53	9.90	516
	TM3	22	85	47.40	8.18	
	TM1	59	112	78.57	7.17	
Maraîchage	TM4	42	85	62.48	11.56	75
	TM3	28	41	34.29	3.44	
	TM1	57	66	61.61	1.71	
Céréaliculture	TM4	54	115	87.62	19.74	37
	TM3	25	30	27.32	1.31	
	TM1	59	66	62.21	1.60	
Brûlis	TM4	38	70	52.14	9.18	43
	TM3	33	62	44.35	8.02	
	TM1	62	91	72.32	8.00	

Jachère	TM4	50	95	59.98	8.54	88
	TM3	30	54	43.90	4.52	
	TM1	61	71	66.98	2.09	
Sable & Sol nu	TM4	59	86	78.38	7.45	21
	TM3	42	85	72.28	12.26	
	TM1	68	99	89.90	8.78	
Ressac	TM4	8	38	13.13	8.56	23
	TM3	14	43	23.22	8.46	
	TM1	58	75	67.35	3.77	
Maquis exposé	TM4	54	60	56.36	1.82	41
	TM3	23	31	27.68	1.82	
	TM1	54	62	59.48	1.73	
Sebkha	TM4	52	62	57.29	2.82	293
	TM3	46	61	57.35	2.90	
	TM1	76	87	82.23	2.07	
Total					1309 pixels	

B 1.4 Techniques d'exploitation d'images satellites [Rich, 1993] [Rich, 1998]

De nombreux domaines ont fait appel aux techniques de l'imagerie. Parmi eux, on peut citer la médecine, la météorologie, etc... L'imagerie est également utilisée dans le domaine militaire pour la localisation des équipements et des troupes.

Pour tirer avantage des données de télédétection, il faut être en mesure d'extraire de l'information significative de l'imagerie.

L'interprétation et l'analyse de l'imagerie de télédétection ont pour but d'identifier et de mesurer différentes cibles dans une image pour pouvoir en extraire l'information utile. Il existe plusieurs fonctions de traitement d'images on peut citer :

1. Prétraitement
2. Rehaussement de l'image
3. Classification et analyse de l'image

B.1.4.1 Pré-traitement (traitement de corrections)

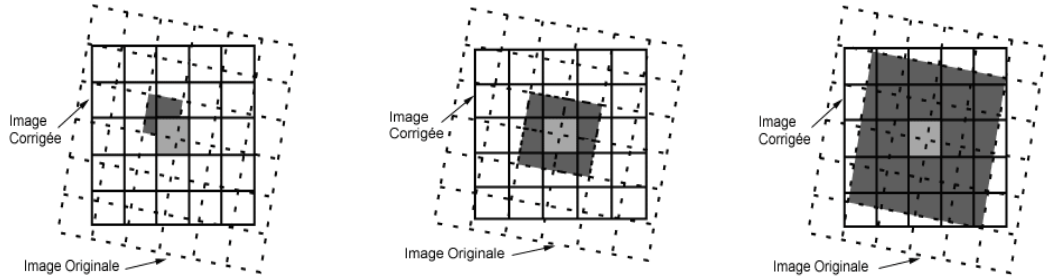
Le Pré-traitement c'est l'ensemble des opérations requises avant l'analyse principale et l'extraction de l'information. Ces opérations se divisent en deux corrections suivantes:

a)- Correction Radio métrique *:*

C'est la correction des données à cause des irrégularités du capteur, des bruits dus au capteur ou à l'atmosphère, et de la conversion des données afin qu'elles puissent représenter précisément le rayonnement réfléchi ou émis mesuré par le capteur.

b)- Correction Géométrique :

Les corrections géométriques comprennent la correction pour les distorsions géométriques dues aux variations de la géométrie Terre-capteur, et la transformation des données en vraies coordonnées (par exemple en latitude et longitude) sur la surface de la Terre (Fig. B.6).



(A) **(B)** **(C)**

Fig.B.6 Correction Géométrique

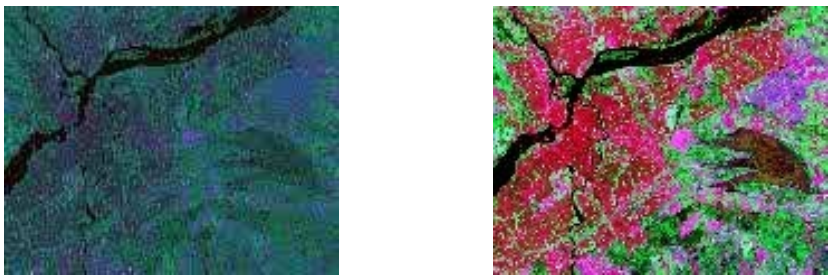
De manière générale, la procédure de correction géométrique comprend les étapes suivantes : a) choix d'un référentiel, b) choix d'un modèle de transformation entre la référence et l'image à corriger, c) application du modèle pour la génération d'une grille rectifiée dans le référentiel choisi, et remplissage de cette grille par une méthode de ré-échantillonnage radiométrique. Durant cette dernière étape, deux cas se présentent :

1. si la référence est choisie comme étant une cartographie géographique, alors l'image corrigée sera géo-référencée par rapport aux coordonnées géographiques au sol et elle gardera sa propre résolution spatiale.

2. si la référence est choisie comme étant une image, alors l'image corrigée sera recalée et superposable à la référence avec la même résolution spatiale de cette dernière.

B.1.4.2 Rehaussement :

Les fonctions de rehaussement ont pour but d'améliorer l'apparence de l'imagerie (Fig. B.7**,** B**)** pour aider l'interprétation et l'analyse visuelles. Les fonctions de rehaussement permettent **l'étirement des contrastes** pour augmenter la distinction des tons entre les différents éléments d'une scène, et le **filtrage spatial** pour rehausser (ou éliminer) les patrons spatiaux spécifiques sur une image.



(A) **(B)**

Fig. B.7 Améliorer l'apparence de l'imagerie

B 2. Bases du Machine Learning :

Les données utilisées dans la deuxième partie dans le chapitre implémentation, sont des données de La Machine Learning Repository. [MLRUCI, 2012]

La Machine Learning Repository est une collection de bases de données, les théories de domaine, et les générateurs de données qui sont utilisées par la communauté d'apprentissage pour l'analyse empirique des algorithmes. L'archive a été créée comme une archive ftp en 1987 par David Aha et les étudiants des cycles supérieurs à l'UC Irvine. Depuis ce temps, il a été largement utilisé par les étudiants, les éducateurs et les chercheurs du monde entier en tant que principale source d'apprentissage automatique des ensembles de données.

La Machine Learning repository est une collection de divers jeux de données réelles.

B 2.1 La base Iris :

Elle doit permettre de reconnaître les différentes sortes d'iris qui existent: la première classe représente les iris setosa, la seconde les iris versicolor, et la dernière les iris virginica. Pour cela, les attributs d'une donnée représentent (dans l'ordre) : la longueur du sépale (en cm), la largeur du sépale (en cm), la longueur du pétale (en cm), la largeur du pétale (en cm).

N est le nombre total d'objets dans le jeu de données, Ni est le nombre de données pour la classe i, et D et C donne la dimension (le nombre d'attributs) et le nombre de classes.

Tableau B.5 Bases du Machine Learning (Iris)

Nom	N	N_i	D	C	Type
Iris	150	50, 50, 50	4	3	Réels

Fig. B .8. La base Iris de la Machine Learning Repository

B 2.2 La base Wine :

Les données ont été utilisées avec beaucoup d'autres pour comparer les différents classificateurs. Les classes sont séparables.

Ces données sont les résultats d'une analyse chimique de vins cultivés dans la même région en Italie, mais découle de trois différents cultivars. L'analyse a déterminé les quantités de 13 constituants dans chacun des trois types de vins. Les attributs sont :

1) L'alcool (Alcohol)
2) L'acide malique (Malicacid)
3) Asch (Ash)
4) alcalinité de cendres (Alcalinity of ash)
5) Magnésium (Magnesium)
6) Total des phénols (Total phenols)
7) Flavonoïdes (Flavanoids)
8) phénols Nonflavanoid (Nonflavanoidphenols)
9) pro anthocyanines (Proanthocyanins)
10) intensité de la couleur (Colorintensity)
11) Hue (Hue)

12) OD280/OD315 des vins dilués (OD280/OD315 of diluted wines)
13) Proline (Proline)

Tableau B.6 Bases du Machine Learning (Wine)

Nom	N	N_i	D	C	Type
Wine	178	59, 71, 48	13	3	Entier, Réels

B 2.3 Les lentilles cornéennes :
Les classes :

1: le patient doit être équipé avec des verres de contact durs,
2: le patient doit être équipé avec des lentilles de contact souples,
3: le patient ne doit pas être pourvue de lentilles de contact.

Renseignements sur les attributs:

a) l'âge du patient: (1) jeune, (2) pré-presbyte, (3) presbytes
b) hypermétrope prescription spectacle: (1) myope, (2) hypermétrope
c) astigmate: (1) non, (2) oui
d) normal déchirer le taux de production: (1) réduite, (2) normale

N est le nombre total d'objets dans le jeu de données, Ni est le nombre de données pour la classe i, et D et C donne la dimension (le nombre d'attributs) et le nombre de classes.

Tableau B.7 Bases du Machine Learning (Les lentilles cornéennes)

Nom	N	N_i	D	C	Type
Les lentilles cornéennes	24	4, 5,15	4	3	Entier

B.3 Données du réseau 30 nœuds

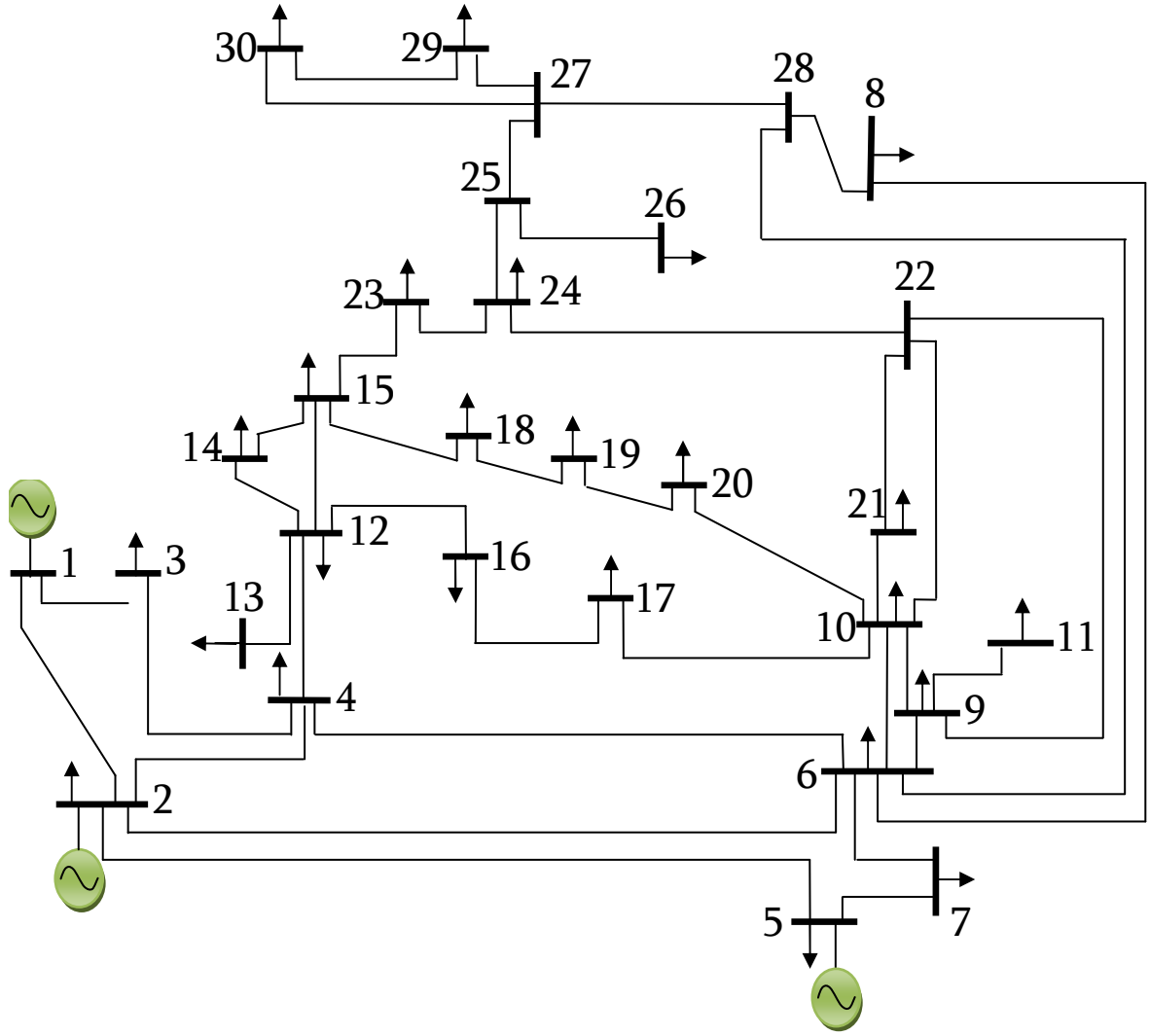


Fig. B.9. Topologie du réseau IEEE-30 nœuds.

La tension de base pour chaque jeu de barres est de 135 kV et la charge totale est de 400 MW pour une puissance de base égale à 100 MVA et les pertes de transmission est de 7MW.

Les valeurs planifiées des puissances et les caractéristiques de réseau sont données par les tableaux B.8 et tableaux B.9.
L'explication des deux opérateurs du dessin ci-dessus:

: générateur

: Charge

Tableau B.8 Les données des charges pour le système d'essai IEEE-30 nœuds.

Nœud N°	**Type du nœud**	**Puissance active [p.u]**	**Puissance réactive [p.u]**
1	bilan	0	0
2	producteur	0.217	0.127
3	consommateur	0.024	0.012
4	"	0.076	0.016
5	producteur	0.942	0.190
6	consommateur	0.000	0.000
7	"	0.228	0.109
8	producteur	0.300	0.300
9	consommateur	0.000	0.000
10	"	0.058	0.020
11	producteur	0.112	0.010
12	consommateur	0.112	0.075
13	producteur	0.013	0.012
14	consommateur	0.062	0.016
15	"	0.082	0.025
16	"	0.035	0.018
17	"	0.090	0.058
18	"	0.032	0.009
19	"	0.095	0.034
20	"	0.022	0.007
21	"	0.175	0.112
22	"	0.000	0.000
23	"	0.032	0.016
24	"	0.087	0.067
25	"	0.000	0.000
26	"	0.035	0.023
27	"	0.000	0.000
28	"	0.000	0.000
29	"	0.024	0.009
30	"	0.106	0.019

Soient R une résistance, X une réactance et Ysh/2 une admittance.

Tableau B.8 Les données des lignes pour le système d'essai IEEE-30 nœuds.

Ligne №	De nœud	Au nœud	R [p.u]	X [p.u]	Ysh/2 [p.u]
1	1	2	0.0192	0.0575	0.0264
2	1	3	0.0452	0.1852	0.0204
3	2	4	0.0570	0.1737	0.0184
4	3	4	0.0132	0.0379	0.0042
5	2	5	0.0472	0.1983	0.0209
6	2	6	0.0581	0.1763	0.0187
7	4	6	0.0119	0.0414	0.0045
8	5	7	0.0460	0.1160	0.0102
9	6	7	0.0267	0.0820	0.0085
10	6	8	0.0120	0.0420	0.0045
11	6	9	0.0000	0.2080	0.0000
12	6	10	0.0000	0.5560	0.0000
13	9	11	0.0000	0.2080	0.0000
14	9	10	0.0000	0.1100	0.0000
15	4	12	0.0000	0.2560	0.0000
16	12	13	0.0000	0.1400	0.0000
17	12	14	0.1231	0.2559	0.0000
18	12	15	0.0662	0.1304	0.0000
19	12	16	0.0945	0.1987	0.0000
20	14	15	0.2210	0.1997	0.0000
21	16	17	0.0824	0.1923	0.0000
22	15	18	0.1070	0.2185	0.0000
23	18	19	0.0639	0.1292	0.0000
24	19	20	0.0340	0.0680	0.0000
25	10	20	0.0936	0.2090	0.0000
26	10	17	0.0324	0.0845	0.0000
27	10	21	0.0348	0.0749	0.0000
28	10	22	0.0727	0.1499	0.0000
29	21	22	0.0116	0.0236	0.0000

30	15	23	0.1000	0.2020	0.0000
31	22	24	0.1150	0.1790	0.0000
32	23	24	0.1320	0.2700	0.0000
33	24	25	0.1885	0.3292	0.0000
34	25	26	0.2544	0.3800	0.0000
35	25	27	0.1093	0.2087	0.0000
36	27	28	0.0000	0.3960	0.0000
37	27	29	0.2198	0.4153	0.0000
38	27	30	0.3202	0.6027	0.0000
39	29	30	0.2399	0.4533	0.0000
40	8	28	0.0636	0.2000	0.0214
41	6	28	0.0169	0.0599	0.0065

Les valeurs des coefficients des fonctions quadratiques de coût des trois générateurs ainsi que les limites minimum et maximum des puissances actives générées sont regroupées dans le tableau B.10:

Tableau B.10 Les données des générateurs pour le système d'essai IEEE-30 nœuds.

Générateur	**Du Nœud**	$F(P_G) = aP_G^2 + bP_G + c \quad [\$/MW]$			P_G [MW]	
		a	**b**	**c**	**Limite minimum**	**Limite maximum**
1	1	0.03546	38.3055	1243.531	35	210
2	2	0.02111	36.3278	1658.559	130	325
3	5	0.01799	38.2704	1356.659	125	315

Annexe C : Méthodes d'optimisation et de classification utilisées pour comparaison avec nos travaux

METHODES D'OPTIMISATION

C 1. Optimisation avec une méthode de point intérieur [Glineur, 1997]

C 1.1. Eléments de base.

Nous sommes à présent en mesure de donner une première description des méthodes de point intérieur, en particulier d'expliciter leur dénomination. Plaçons-nous dans le cas du programme linéaire dans sa forme standard :

$$\begin{aligned} &\min c^T x, \quad x \in R^n \\ &\text{avec } Ax = b \text{ et } x \geq 0 \end{aligned} \qquad \text{(Eq.C.1)}$$

Les méthodes de point intérieur partent d'un point situé *à l'intérieur* du polyèdre admissible. Pour être plus précis, ces méthodes nécessitent généralement un point de départ *strictement* intérieur, soit $x \in \Im_0$ avec les définitions suivantes :

$\Im = \{x \mid Ax = b \text{ et } x \geq 0\}$ $\Im_0 = \{x \mid Ax = b \text{ et } x > 0\}$

Le principe des méthodes de point intérieur consiste alors à faire évoluer *de façon itérative* ce point vers une solution optimale du problème, tout en restant à l'intérieur strict ($\Im_0$) du polyèdre admissible. En d'autres termes, on construit une suite x_0, x_1, x_2, ..., x_i, ... d'itérés intérieurs convergeant vers une solution, comme illustré sur la Figure (**Fig. C 1.)**.

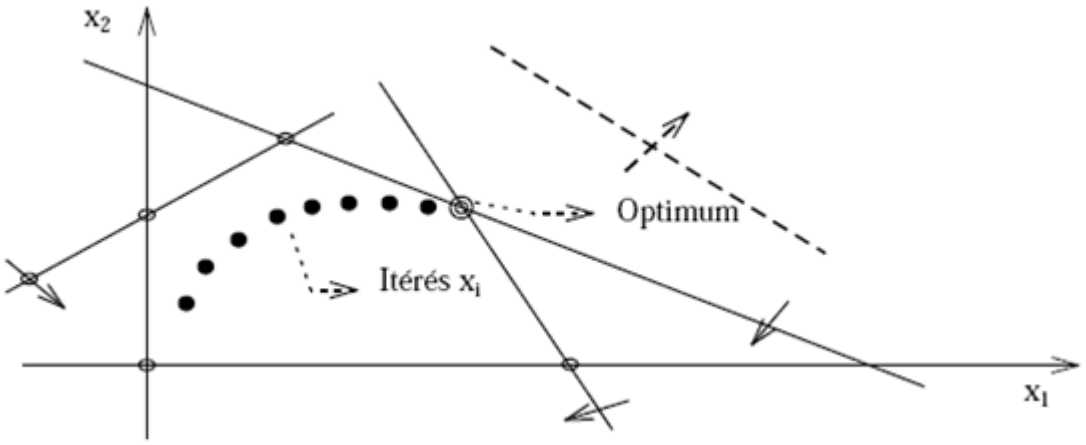


Fig. C.1 Convergence des itérés d'une méthode de point intérieur

La complexité algorithmique de ces méthodes : elles sont de type polynomial.

C 1.2. Méthode de Newton :

Pour rappel, sot le problème suivant à résoudre : trouver $\mathbf{x} \in \mathbf{R}^n$ tel que $\mathbf{f(x) = 0}$ où f est une fonction de $\mathbf{R}^n \rightarrow \mathbf{R}^m$. la méthode de Newton consiste alors à passer de l'itéré $\mathbf{x_k}$ à $\mathbf{x_{k+1}}+\mathbf{\Delta x_k}$ avec $\mathbf{\Delta x_k = -J(x_k)^{-1}\, f(x_k)}$. Plutôt qu'inverser directement le jacobien **J** de la fonction **f**, on préfère généralement calculer $\mathbf{\Delta x_k}$ comme la solution du système linéaire $\mathbf{J(x_k)\, \Delta x_k = -\, f(x_k)}$

Géométriquement, cette méthode fournit comme itéré suivant le zéro de l'approximation locale du premier ordre (c'est-à-dire l'hyperplan tangent) de la fonction f autour de l'itéré courant. La figure (**Fig. C 2.)** explique la situation dans le cas unidimensionnel.

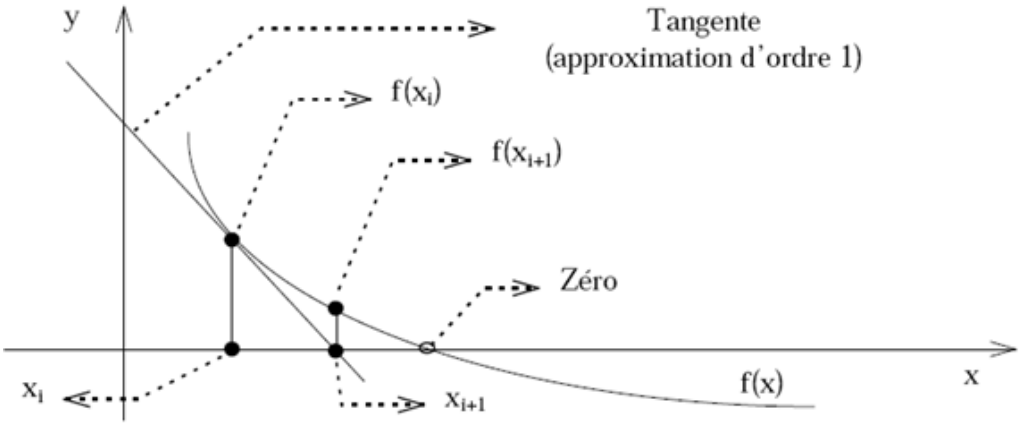


Fig. C.2 Une itération de la méthode de Newton pour une fonction scalaire

C 1.3. Modèle d'algorithme de point intérieur

La méthode de résolution naïve suivante

Algorithme C.1 : Méthode de point intérieur

Soit une suite $\tau_1, \tau_2, \tau_3,\ldots$ décroissante et tendent vers zéro (x_0, y_0, s_0) un itéré initial

Pour k=1,2,3,… **faire**

- Appliquer la méthode de Newton à la résolution des problèmes primal et dual avec le paramètre τ_k pour le terme barrière. Ceci se fait par une succession de pas définie par (NWT), en partant de la solution du k précédent $(x_{k-1}, y_{k-1}, s_{k-1})$
- (x_k, y_k, s_k) est la solution fournie par la méthode de Newton

Fin Pour

On peut raisonnablement s'attendre à ce que l'itéré (x_k,y_k,s_k) converge vers l'optimum cherché.

C 2. Optimisation avec les algorithmes évolutionniste

Les algorithmes évolutionniste sont basés sur des principes simples. En effet, peu de connaissance sur la manière de résoudre ces problèmes sont efficaces l'évolution. C'est pourquoi dans nombreux domaines, les chercheurs ont été amenés à s'y intéresser. Les algorithmes évolutionniste sont une classe d'algorithme d'optimisation par recherche Probabiliste basés sur le modèle de l'évolution naturelle. Ils modalisent une population d'individus par des point dans un espace. Ils ont montré leur capacité à éviter la convergence. Des solutions vers des optima locaux.

Plusieurs types dévolutions ont été développés, donnant naissance à quatre grandes tendances : les stratégies d'évolution, la programmation évolutive, la programmation génétique et les algorithmes génétiques.

C 2.1. Les algorithmes génétiques

C 2.1.1 Principes généraux

Les Algorithme Génétique (AG) font partie d'une famille de méthode stochastique appelés méthode évolutionniste qui reposant sur une analogie avec la théorie de l'évolution naturelle, selon laquelle les individus d'une population les mieux adapté à leur environnement ont une plus grande probabilité de survivre de se reproduire de génération en génération, en donnent des descendent encore mieux adaptés. Depuis une trentaine d'années d'intérêt pour les algorithmes génétique va croissant en raisons de leurs nombreux avantages sur les autres technique d'optimisation : ils sont robustes, rapides, Suffisante généraux pour pouvoir s'adapter à n grand nombre de situation et enfin ne demandent aucun connaissance précise sur le system a optimisé.

Les algorithmes génétiques ont été développés dans les années 70 par Holand puis approfondis par Goldberg, ils sont certainement la branche des Algorithme évolutionnistes les plus connue et les plus utilisé .La particularité de ces algorithme est le fait qu'ils font évaluer des populations d'individus codés par une chaîne binaire. Ils utilisant l'opérateur de mutation

et de recombinaison de différents types. Le but d'un algorithme génétique est d'optimisé une fonction donnée dans un espace de recherche précis. Dans le cas général, un algorithme génétique a besoin de quatre composants fondamentaux : Une fonction de codage qui transforme les données de l'espace de recherche en données utilisable par un ordinateur : par exemple une séquence de bits ou bien un nombre réel. Un moyen de créer une population initiale à partir des solutions potentielles.

Une fonction qui permet d'évaluer l'adaptation d'un chromosome, ce qui offre la possibilité de comparer les individus. Cette fonction sera en fait construite à partir du critère que l'on désire optimiser. L'application de cette fonction à un élément de la population donnera sa performance (évaluation). Des opérateurs qui altèrent les enfants après la reproduction. On choisit une population initiale de taille n, c'est-à-dire que l'on tire au hasard le plus uniformément possible un certaine nombre d'éléments qui seront appelés chromosome dans l'espace des données. A fin de réaliser l'analogie avec la génétique, il faut disposer comme nous l'avons vu d'opérateurs de sélection et de recombinaison qui vont permettre à cette population dévaluer.

L'utilisation de ces trois opérateur permet de conserve une population bien diversifiée (c'est-à-dire bien répartie dans l'espace) et par conséquent d'accéder à tout l'espace de recherche. [Yann-Chang, 1996] [Basaputer, 2003] [Ali Hassan, 2003].

Algorithme C.2. Algorithme génétique

```
Initialisation
Evaluation de la population
Tantque Test d'arrêt n'est pas vérifié faire
    Sélection, Croisement, Mutation
    Evaluation des nouveaux individus
    Remplacement des anciens individus
FinTantque
Solution Final
```

C 3. Optimisation par la méthode Parallel Asynchronous PSO :

La méthode Particle Swarm Optimization (PSO) (Optimisation avec les essaims de particules) est une méthode d'optimisation stochastique, pour les fonctions non-linéaires, basée sur la reproduction d'un comportement social et développée par le Dr. EBERHART et le Dr. KENNEDY **[**Russel, 2001] [Ricardo, 2007] en 1995.

L'origine de cette méthode vient des observations faites lors des simulations informatiques de vols groupés d'oiseaux et de déplacements de bancs de poissons [Heppner, 1990]**.** Ces simulations ont mis en valeur la capacité des individus d'un groupe en mouvement à conserver une distance optimale entre eux et à suivre un mouvement global par rapport aux mouvements locaux de leur voisinage.

D'autre part, ces simulations ont également révélé l'importance du mimétisme dans la compétition qui oppose les individus à la recherche de la nourriture. En effet, les individus sont à la recherche de sources de nourriture qui sont dispersées de façon aléatoire dans un

espace de recherche, et dès qu'un individu localise une source de nourriture, les autres individus vont alors chercher à l'imiter.

Ce comportement social basé sur l'analyse de l'environnement et du voisinage constitue alors une méthode de recherche d'optimum par l'observation des tendances des individus voisins. Chaque individu cherche à optimiser ses chances en suivant une tendance qu'il modère par ses propres vécus.

C 3.1. Formalisation

Un essaim de particules est caractérisé par :

- **le nombre de particules** de l'essaim, noté nb ;
- **la vitesse maximale** d'une particule, notée $\vec{V}_{max}$
- **la topologie et la taille du voisinage** d'une particule qui définissent son réseau social.
- **l'inertie** d'une particule, notée Ψ ;
- **les coefficients de confiance**, notés $\rho_1$et ρ_2, qui pondèrent le comportement conservateur (c'est à dire la tendance à retourner vers la meilleure solution visitée) et le panurgisme (c'est à dire la tendance à suivre le voisinage).

Une particule est caractérisée, à l'instant *t*, par :

$\vec{x}_i(t)$: Sa position dans l'espace de recherche ;
$\vec{V}_i(t)$: Sa vitesse ;
$\vec{x}_{pbest_i}$: La position de la meilleure solution par laquelle elle est passée ;
$\vec{x}_{vbest_i}$: La position de la meilleure solution connue de son voisinage ;
$pbest_i$: La valeur de fitness de sa meilleure solution ;
$vbest_i$: La valeur de fitness de la meilleure solution connu du voisinage ;

C 3.2. Configuration de la méthode

C 3.2.1. Nombre de particules

La quantité de particules allouées à la résolution du problème dépend essentiellement de deux paramètres :

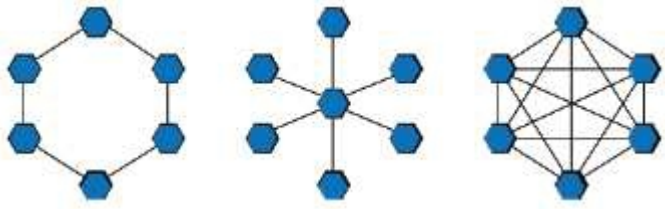
La taille de l'espace de recherche et le rapport entre les capacités de calcul de la machine et le temps maximum de recherche. Il n'y a pas de règle pour déterminer ce paramètre, faire de nombreux essais permet de se doter de l'expérience nécessaire à l'appréhension de ce paramètre.

C 3.2.2. Topologie du voisinage

La topologie du voisinage défini avec qui chacune des particules va pouvoir communiquer. Il existe de nombreuses combinaisons dont les suivantes sont les plus utilisées:

1. **Topologie en anneau :** chaque particule est reliée à $n$particules (en général,$n = 3$), c'est la topologie la plus utilisée.
2. **Topologie en rayon :** les particules ne communiquent qu'avec une seule particule centrale.

3. **Topologie en étoile :** chaque particule est reliée à toutes les autres, c'est à dire l'optimum du voisinage est l'optimum global ;



Anneau ($\mathbf{n = 2}$) Rayon Etoile

Fig. C.3 Topologie du voisinage.

Le voisinage géographique auquel nous sommes amenés à penser en premier lieu n'est pas nécessairement pertinent car, d'une part, il s'agirait d'un voisinage trop local, et d'autre part car la sociabilisassions des particules tend à rendre tout voisinage social en voisinage géographique. Enfin, c'est un voisinage très lourd en terme de calculs car nécessitant de recalculer le voisinage de chaque particule à chaque itération.

C 3.2.3. Coefficients de confiance

Les variables de confiance pondèrent les tendances de la particule à vouloir suivre son instinct de conservation ou son panurgisme. Les variables aléatoires ρ_1 et ρ_2 peuvent être définies de la façon suivante :

$$\begin{cases} \rho_1 = r_1.c_1 \\ \rho_2 = r_2.c_2 \end{cases}$$

Où r_1 et r_2 suivent une loi uniforme sur $[0,1]$ et c_1 et c_2 sont des constantes positives déterminées de façon empirique et suivant la relation $c_1 + c_2 \leq 4$

C 3.2.4. Vitesse maximale et coefficient de construction

Afin d'éviter que les particules ne se déplacent trop rapidement dans l'espace de recherche, passant éventuellement à côté de l'optimum, il peut être nécessaire de fixer une vitesse maximale (notée $\vec{V}_{max}$) pour améliorer la convergence de l'algorithme.

Cependant, on peut s'en passer si on utilise un coefficient de construction k(introduit par Maurice CLERC **[Clerc, 2002]**) et qui permet de resserrer l'hyper-espace de recherche.

L'équation de la vitesse devient alors :

$$k = 1 - \frac{1}{\rho} + \frac{\sqrt{|\rho^2 - 4\rho|}}{2} \qquad \text{(Eq.C.2)}$$

Avec :

$$\rho = \rho_1 + \rho_2 > 4 \qquad \text{(Eq.C.3)}$$

$$\vec{V}_i(t) = k.\left(\vec{V}_i(t-1) + \rho_1.\left(\vec{x}_{pbest_i} - \vec{x}_i(t)\right) + \rho_2.\left(\vec{x}_{vbest_i} - \vec{x}_i(t)\right)\right) \qquad \text{(Eq.C.4)}$$

Les études de SHI et EBERHART indiquent que l'utilisation d'un coefficient de construction donne généralement un meilleur taux de convergence sans avoir à fixer de vitesse maximale. Cependant, dans certains cas, le coefficient de construction seul ne permet pas la convergence vers la solution optimale pour un nombre d'itérations donné. Pour résoudre ce problème, il peut être intéressant de fixer $\vec{V}_{max} = \vec{x}_{max}$ en plus du coefficient de construction, ce qui, selon les études de SHI et EBERHART, permet d'améliorer les performances globales de l'algorithme.

C 3.2.5. Facteur d'inertie

Le facteur d'inertie Ψ (introduit par SHI et EBERHART) permet de définir la capacité d'exploration de chaque particule en vue d'améliorer la convergence de la méthode. Une grande valeur de $\Psi\ (>1)$ est synonyme d'une grande amplitude de mouvement et donc, d'exploration globale. Au contraire, une faible valeur de $\Psi\ (<1)$ est synonyme de faible amplitude de mouvement et donc, d'exploration locale. Fixer ce facteur, revient donc à trouver un compromis entre l'exploration locale et l'exploration globale.

Le calcul de la vitesse est alors défini par :

$$\vec{V}_i(t) = \Psi.\vec{V}_i(t-1) + \rho_1.\left(\vec{x}_{pbest_i} - \vec{x}_i(t)\right) + \rho_2.\left(\vec{x}_{vbest_i} - \vec{x}_i(t)\right) \qquad \text{(Eq.C.5)}$$

La taille du facteur d'inertie influence directement la taille de l'hyper-espace exploré et aucune valeur de Ψ ne peut garantir la convergence vers la solution optimale.

Les études menées par SHI et EBERHART indiquent une meilleure convergence pour $\psi \in [0.8,1.2]$. Au delà de 1.2, l'algorithme tend à avoir certaines difficultés à converger **[Dorigo, 1999b]**.

C 3.2.6. Initialisation de l'essaim

La position des particules ainsi que leur vitesse initiale doivent être initialisés aléatoirement selon une loi uniforme sur [0,1].

C 3.2.7. Critères d'arrêt

Comme indiqué précédemment, la convergence vers la solution optimale globale n'est pas garantie dans tous les cas de figure même si les expériences dénotent la grande performance de la méthode. De ce fait, il est fortement conseillé de doter l'algorithme d'une porte de sortie en définissant un nombre maximum d'itérations (que nous noterons $nbIter_{max}$).

L'algorithme doit alors s'exécuter tant que l'un des critères de convergence suivant n'a pas été atteint :

- $nbIter_{max}$ a été atteint ;
- la variation de la vitesse est proche de 0 ;
- le fitness de la solution est suffisant.

C 3.3 Parallel Asynchronous PSO (PAPSO)

Tandis que plusieurs modifications à l'algorithme original PSO ont été faites pour augmenter la robustesse et la puissance de calcul, un des principaux problèmes est de savoir si une synchrone ou asynchrone approche est utilisée pour mettre à jour les positions et les vitesses des particules. L'algorithme séquentiel synchrone PSO fait la mis à jour à la fin de chaque itération d'optimisation, par contre l'algorithme asynchrone met à jour les positions et les vitesses continuellement basées sur l'information disponible actuellement.

Algorithme C.3 Pseudo-code de l'algorithme de PAPSO.

```
//Initialiser l'optimisation
    Initialiser les constants d'algorithme
    Initialiser aléatoirement touts les positions et  les vitesses des particules
//Exécuter l'optimisation
    Pour k = 1 jusqu'à le nombre d'itérations
        Pour i = 1 jusqu'à le nombre de particules
            Evaluer l'analyse de la fonction F(x_k^i)
            Convergence de contrôle
            Mettre à jour x_pbest_i , x_vbest_i et les positions et les vitesses x_i et V_i.
        Fin pour
    Fin pour
    Résultat
```

METHODES DE CLASSIFICATION

C 4. La méthode des K-moyennes [Monm,2000a][MacQueen,1967][Fekraoui,2005] [Emanuel,2000]

La méthode des K-moyennes est une méthode de classification géométrique bien adaptée aux espaces vectoriels de grande dimension.

Elle est régulièrement utilisée pour effectuer des classifications non supervisées d'images multispectrales.

L'algorithme des K-moyennes s'efforce de trouver les centroïdes les plus représentatifs de nuages de points.

Algorithme C.4. Algorithme des k-moyennes

Algorithme :

- Initialisation : choisir les centres initiaux des classes (choix aléatoire)
- Affectation : affecter chaque point du nuage au groupe dont le centre est le plus proche
- Mise à jour des centres : calculer les nouveaux centres (la moyenne des nouveaux points de la classe)
- Test de convergence :

- soit le nombre d'itérations
- soit les centroïdes sont inchangés

L'avantage de cet algorithme est avant tout leur grande simplicité mais aussi leur complexité algorithmique qui reste raisonnable *O(NKT)* où *K* est le nombre de classes, *T* est le nombre d'itérations effectuées et *N* est le nombre d'objets. On peut montrer que d'une itération à l'autre (i.e. d'une partition à l'autre), l'inertie intraclasse $\zeta_{int\,ra}$ déroît ce qui entraine la convergence de l'algorithme [Jain, 1988] ;

Cependant ces méthodes souffrent de certains inconvénients : d'une part, le calcul de moyenne qu'elles utilisent est très sensible aux points aberrants et restreint leurs applications aux données numériques.

D'autre part, la partition finale obtenue est très dépendante du choix des centres initiaux.

C 5. La méthodes de nuées dynamiques (*Migrating Means*) aussi appelée *ISODATA*[Takah, 1995] [Fekraoui, 2005] [Emanuel,2000].

La méthode des nuées dynamiques largement développé par Diday dans [Diday, 19 71] se distingue principalement de l'approche précédente par le mode de représentation des classes appelé aussi noyau.
Ce dernier peut être son centre de gravité (dans ce cas nous retrouvons l'approche des centres mobiles).

L'algorithme des nuées dynamiques cherche à optimiser un critère objectif mesurant l'adéquation entre une partition et un mode de représentation des classes de cette partition. En pratique, l'algorithme converge lorsque ce critère à optimiser cesse de décroitre de façon sensible, ou lorsqu'un nombre fixé d'itération est atteint. Celeux et al. Précisent dans [Celeux, 1989] que : « le problème d'optimisation dans ce cas se pose en terme de recherche simultanée de la classification et de la représentation des classes de cette classification parmi un ensemble de classifications et de représentations possibles, qui minimisent le critère fixé ». Afin de minimiser ce critère, l'algorithme des nuées dynamiques utilise principalement une étape de représentation suivie d'une étape d'affectation de façon itérative jusqu'à jusqu'à la convergence qui donne une solution localement optimale au problème posé.

La méthode des nuées dynamiques comme les approches précédentes de classificati on automatique par partitionnement fournit une solution dépendante de la configuration initiale qui est généralement faite par tirage au hasard.

Algorithme C.5 Algorithme des ISODATA

- C’est le même algorithme que le K_means.
- Mais cherche à équilibrer les classes
- Fusion de deux classes (diminution du nombre de classes) si la distance inter centre est trop faible

Bibliographie

Bibliographie

[Abderrahmani, 2011] Abderrahmani Abdesselam, & Nasri Mohamed . Economic emission dispatch solution using parallel synchronous PSO algorithms. Acta Electrotechnica et Informatica, Vol. 11, No. 1, 2011, 66–70

[Alhanjouri, 2011] Alhanjouri Mohammed & Belal Alfarra. " Ant Colony versus Genetic Algorithm based on Travelling Salesman Problem". International Journal of Computer Technology and Applications, Vol 2, No. 3, pp. 570-578, May-June 2011, ISSN:2229-6093.

[Ali, 1997] Ali, S. et Zimmer, R. (1997). The question concerning emergence. In Proceedings of CASYS97, pages 22-26.

[Ali Hassan, 2003] Ali Hassan, AI-Mohammed and Ibrahim Elamin"Capacitor Placement In Distribution Systems Using Artificial Intelligent Technique ", Paper accepted for presentation at 2003 IEEE Bologna Power Tech Conference, June 23-23, Bologna, Italy.

[Allaoua , 2009] Alloua B. & Laoufi A., « Optimal Power Flow Solution Using Ant Manners for Electrical Network », Advances in Electrical and Computer Engineering, Vol. 9, N°. 1, 2009.

[Arkin, 1998] Arkin, R., & Balch, T. (1998). Artificial Intelligence and Mobile Robots, Chapter Cooperative Multiagent Robotic System, pp. 277-296. MIT Press, Cambridge, Massachusetts.

[Andrey, 1998] P. Andrey, et P. Tarroux,. Unsupervised segmentation of Markov random field modeled textured images using selectionist relaxation, IEEE Transactions on Pattern Analysis and Machine Intelligence, 3(20): 252–262, 1998.

[Azza, 2004] Azzag H., Picarougne F., Guinot C., & Venturini G. (2004). Un Survol des Algorithmes Biomimétiques pour la Classification. Classification et Fouille de Données. RNTI-C-1, Cépaduès, pp. 13-24.

[Azzag, 2004a] H. Azzag, C. Guinot et G. Venturini, . AntTree: web document clustering using artificial ants. Proceedings of the 16th European Conference on Artificial Intelligence (ECAI 04), p. 480-484, IOS Press, Valencia, Spain 2004.

[Azzag, 2004b] Hanéne Azzag, Gilles Venturini, « A clustring model using artificial ants », Université François-Rabelais, Tours - France, 2004.

[Benasla, 2008] Benasla Lahouaria, Belmadani Abderrahim and Rahli Mustapha. Application of SUMT Method to Combine Economic and Emission Dispatch. Leonardo Journal of Sciences ISSN 1583-0233 Issue 13, July-December 2008 p. 122-132

[Brucker, 1978] P. Brucker. On the complexity of clustering problems. Optimierung und Operations Research, In R. Henn, B. Korte and W. Oletti (Eds). Lecture notes in Economics and Mathematical Systems, 1978.

[Bandyopadhyay, 2002] S. Bandyopadhyay et U. Maulik Genetic clustering for automatic evolution of clusters and application to image classification, Pattern Recognition, vol.35, no. 6, pp. 1197-1208, 2002.

[Basaputer, 2003] Basaputer.P and Ongsakul.W"Optimal Power Flow with Mulit-type FACTS Devices by Hybrid TS/SA Approach " IEEE ICIT 02, Bangokok, THAILAND, pp 285-290. Date of Conference: 25-28 May 2003

[Beckers, 1994] R. Beckers, , O.,Holland, and J. Deneubourg, From local actions to global tasks: stigmergy and collective robotics . In Proceedings of Artificial Life 4, pages 181–189.

MIT Press. 1994

[Belh, 1998] Belhadj A. (1998). Contribution au Logiciel d'Analyse et de Traitement d'Images Satellitaires (L.A.T.I.S). Analyse Spatiale-Spectrale d'Images Satellitaires Appliquées à la Cartographie Thématique. Thèse de Doctorat d'état en électronique, spécialité Traitement d'Images et Télédétection, USTHB, Alger, 268 p

[Bellet, 1998] F.Bellet, Une approche incrémentale, coopérative et adaptative pour la segmentation des images en niveau de gris. Thèse de doctorat, Institut National Polytechnique de Grenoble, France, juin 1998.

[Berro, 2001]Alain Berro, "Optimisation Multiobjectif et Stratégie d'évolution en environnement Dynamique " thèse de Doctorat ; 18 Décembre 2001 ;Université des sciences sociales ToulouseI ; pp 14, 27, 29.

[Benya, 2008] Benyamina A. (2008). Conception d'une approche Hybride pour une classification multi source de données de télédétection. Thèse de magistère de l'université de Béchar. Discipline : Informatique.

[Benyamina, 2011] A.Benyamina and H.Fizazi, 2011. The efficiency of the adapted AntClust algorithm for satellite images clustering. Malaysian Journal of Computer Science. Issn 0127-9084.

[Bhandarkar, 1999] S. M. Bhandarkar et H. Zhang, Image Segmentation Using Evolutionary Computation, IEEE Trans. On Evolutionary Comp., Vol. 3, No. 1, pp. 1-21, April, 1999

[Bhanu,1995] B. Bhanu, S.Lee, , and J. Ming, Adaptive image segmentation using a genetic algorithm. IEEE Trans. Systems Man Cybernet. 25 (1995), pp. 1543–1567.

[Bonab, 1999] Bonabeau E. , Dorigo M., Therauluz G. Swarm Intelligence. From Natural to Artificial Systems. Oxford University Press, New York, 1999.

[Bonabeau, 1997] E Bonabeau et G. Theraulaz G., Auto-organisation et comportements collectifs : la modélisation des sociétés d'insectes, Auto-organisation et comportement, Editions Hermès, 1997.

[Bonabeau, 1999] E. Bonabeau, M. Dorigo, G.Theraulaz, Swarm Intelligence : From Natural to Artificiel Systems, NEW York, Oxford University Press,1999.

[Bonabeau, 2000] E Bonabeau et G. Theraulaz G., L'intelligence en essaim, pour la science, 282 (3): pp. 66-73, N° 271 mai 2000.

[Bullnheimer, 1997] B. Bullnheimer, R.Hartl, et C. Strauss, Applying the Ant System to the Vehicle Routing Problems, 2nd Metaheuristic International Conference (MIC-97), Sophia-Antipolis, France.1997;

[Bullnheimer, 1999] B. Bullnheimer, R.F. Hartl, and C. Strauss. An improved ant system algorithm for the vehicle routing problem. *Annals of Operations Research*, 89 :319–328, 1999.

[Camazine, 2001] S. Camazine, J. L. Deneubourg, N. R. Francks, J. Sneyd, G. Theraulaz, and E. Bonabeau. Self-Organization in Biological Systems. Princeton University Press and Oxford, 2001.

[Camazine, 2002] S. Camazine, J-L. Deneubourg , N.R. Franks , J. Sneyd , G. Theraulaz et E. Bonabeau. Self-Organization in Biological Systems, Princeton University Press, 2002.

[Candillier, 2006] Laurent Candillier, Isabelle Tellier, Fabien Torre, Olivier Bousquet. Évaluation en cascade d'algorithmes de clustering. Université Charles de Gaulle - Lille 3, 2006.

[Celeux, 1989] Celeux G., Diday E., Lechevallier Y., Govaert G. and Ralambondrainy H. Classification automatique des données, Editions Dunod, Paris, 1989.

[Cell, 1989] Celleux G., Diday E., Govaert G., Lechevallier Y. & Ralambondrainy H. (1989). Classification automatique des données. Editions Dunod Informatique, Paris, France.
[Charles, 2004] Christophe CHARLES, « SearchXQ: une méthode d'aide à la navigation fondée sur O-means, algorithme de classification non-supervisée. Application sur un corpus juridique Français », thèse pour obtenir le grade de docteur de l'Ecole des Mines de Paris, spécialité « Informatique Temps Réel, Robotique et Automatique », 17 décembre 2004.
[Chi, 2006]Chi S. and Yang C. Integration of Ant Colony SOM and k-means for Clustering Analysis. Knowledge Based Intelligent Information and Engineering Systems, LNCS, Springer, Vol. 4251, 2006, pp. 1-8.
[Chopard, 2006] Chopard B. Méthodes et Heuristiques d'Apprentissage et d'optimisation. Cours Université de Genève. septembre 2006.
[Christophe, 2001] Christophe SAINT-JEAN. Classification paramétrique robuste partiellement supervisée en reconnaissance des formes.
Thèse de doctorat. Spécialité Informatique, Université de La Rochelle UFR Sciences Laboratoire d'Informatique et d'Imagerie Industrielle. **[Ciobanu, 2001]** Ciobanu C. & Marin G., « On Heuristic Optimization », An. St. Univ. Ovidius Constanta, Vol.9, pp 17-30, 2001.
[Collette, 2002] Collette Y. & Siarry P.,« Optimisation multi-objectif », Eyrolles édition, 2002.
[Colorni, 1991] Colorni A., Dorigo M. et Maniezzo V. Distributed Optimization by Ant Colonies, actes de la première conférence européenne sur la vie artificielle, Paris, France, Elsevier Publishing, 134-142, 1991.
[Colorni, 1992] A. Colorni, M. Dorigo, V. Maniezzo. Distributed optimization by ant colonies, in Toward a practice of autonomous systems : proceedings of the first European Conference on Artificial Life (ECAL 91). MIT Press, 1992, p. 134-142.

[Costanzo, 2006] COSTANZO Andrea. LUONG Thé Van. MARILL Guillaume. Optimisation par colonies de fourmis. Université de nice- Sophia Antipolis. 19 mai 2006
[Costa, 1997] D. Costa. et A. Hertz., Ants Can Color Graphs, Journal of the Operational Research Society, (48): p. 295-305. 1997
[Cucc, 1993] Cucchiara, R. (1993). Analysis and comparison of different genetic models for the clustering problem in image analysis. In Albrecht, R., Reeves, C., & Steele, N. editors, International Conference on Artificial Neural Networks and Genetic Algorithms, pp. 423-427. Spring-Verlag.
[Dasgupta, 1997] Dasgupta, D. and Attoh-Okine, N. (1997). Immune-based systems : A survey. In Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, volume 1, pages 369-374, Orlando. IEEE Press.
[Dasgupta, 1999] Dasgupta, D. (1999). Artificial Immune Systems and their applications. Springer Verlag.
[David, 2010] David M., Bart B. & Tom F. (2010)**.** Editorial Survey : Swarm Intelligence for Data Mining. Machine Learning Manuscript.
[De Castro, 1999] De Castro, L. and Von Zuben, F. (1999). Artificial Immune Systems : Part I : Basic Theory and Applications. Technical Report TR-DCA 01/99, Department of Computer Engineering and Industrial Automation, School of Electrical and Computer Engineering, State University of Campinas, Brazil.
[De Castro, 2000] De Castro, L. and Von Zuben, F. (2000). Artificial Immune Systems : Part II - A Survey of Applications. Technical Report DCA-RT 02/00, Department of Computer Engineering and Industrial Automation, School of Electrical and Computer Engineering, State University of Campinas, Brazil.

[Deb, 1998] Dep, K. (1998). Genetic algorithm in search and optimization: the technique and applications. Proceedings of International Workshop on Soft Computing and Intelligent Systems, Calcutta, India: Machine Intelligence Unit, Indian Statistical Institute, pp. 58-87.

[DeCa, 2002] De Castro, L. N., & Timmis, J. I. (2002). Artificial Immune Systems: A New Computational Intelligence Approach. Springer-Verlag.

[Deneub, 1990] Deneubourg J. L., Goss S., Franks N., Sendova-Franks A., Detrain C. & Chretien L. (1990). The dynamic of collective sorting robot-like ants and ant-like robots. Proceedings of the first Conference on Simulation of Adaptive Behaviours.

[Deneubourg, 1990] J.L Deneubourg, S Aron, S. Goss, et J.M Pasteels, The self-organizing exploratory pattern of the argentine ant. Dans Journalon insect Behavior, 3: 159-168, 1990.

[Deneubourg, 1991] Deneubourg, J. L., Goss, S., Franks, N., Sendova-Franks, A., Detrain, C. and Chretien, L. [1991]. «The dynamic of collective sorting robot-like ants and ant-like robots», in J. A. Meyer and S. W. Wilson (eds), SAB 90 – 1st Conference On Simulation of Adaptive Behavior: From Animals to Animats, MIT Press.

[Di Caro, 1998] Di Caro, G., Dorigo, M.(1998), AntNet: Distributed Stigmergetic Control for Communications Networks, Journal of Artificial Intelligence Research (JAIR), (9): p.317-365.

[Dorigo, 1991] M. Dorigo, V. Maniezzo & A. Colorni (1991). Positive feedback as a search strategy, Technical Report 91-16 Politecnico di Milano, Italy.

[Dorigo, 1991] M. Dorigo, V. Maniezzo, A. Colorni, Positive feedback as a search strategy, Technical Report 91-16 Politecnico di Milano, Italy.1991.

[Dorigo, 1992] Dorigo M., Optimization, Learning and Natural Algorithms, PhD thesis, Politecnico di Milano, Italie, 1992

[Dorigo, 1995] Dorigo, M., 1995and Gambardella, L. M., Ant-Q: A Reinforcement Learning approach to the traveling salesman problem, in: *Proceedings of ML-95, Twelfth International Conference on Machine Learning*, A. Prieditis and S. Russell (eds.) (Morgan Kaufmann, San Francisco, CA) pp. 252–260.

[Dorigo, 1996] M. Dorigo, V. Maniezzo, A. Colorni,, The Ant System: Optimization by a Colony of Cooperating Agents, IEEE Transactions on Systems, Man and Cybernetics-Part B, 1(26): p. 29-41. 1996.

[Dorigo, 1997a] M. Dorigo, M. Gambardella, M. Ant Colony for the Traveling Salesman Problem, BioSystems, (43): 73-81, 1997.

[Dorigo, 1997b] M. Dorigo, M. Gambardella, Ant Colony System: A Cooperative Learning Aproach to the Traveling Salesman Problem, IEEE Transactions on Evolutionary Computation Vol 1: p. 53-66 1997.

[Dorigo, 1999a] M. Dorigo, G. Di Caro, and L.M. Gambardella. Ant algorithms for discrete optimization. *Artificial Life*,5(2) :137–172, 1999.

[Dorigo, 1999b] M. Dorigo and G. Di Caro. The Ant Colony Optimization meta-heuristic. In D. Corne, M. Dorigo, and F. Glover, editors, *New Ideas in Optimization*, pages 11–32.McGraw Hill, UK, 1999.

[Dorigo, 1999c] M. Dorigo, G. Di Caro. Ant colony optimization: a new meta-heuristic. Proceedings of the Congress on Evolutionary Computation. 1999.

[Dorigo, 2000] Dorigo, M., Bonabeau, E., & Theraulaz, G. (2000). Ant algorithms and stygmergy. Futur Generation Computer System, pp. 851-871.

[Dorigo, 2004] M .Dorigo, T. STUZLE,. Ant Colony Optimization. MIT Press, Cambridge, MA, 2004.

[Dorigo, 1992] M. Dorigo. Optimization, Learning and Natural Algorithms (in Italian). PhD thesis, Dipartimento di Elettronica, Politecnico di Milano, Italy, 1992.

[Dorigo, 2003] Dorigo, M. and Stützle, T. (2003). Handbook of Metaheuristics, volume 57 of International series in operations research and management science, chapter The Ant Colony Optimization Metaheuristics : Algorithms, Applications and Advances. Kluwer Academic Publishers, Boston Hardbound.

[Dorigo, 2006] Dorigo M., Birattari M., Stützle T. *Ant Colony Optimization : Artificial Ants as a Computational Intelligence Technique*. IEEE Computational Intelligence Magazine, volume 1, numéro 4, pages 28-39, 2006.

[Dréo, 2003] Dreo Johann., Pétrowski A., Siarry P. & Eric T. (2003). Métheuristiques pour l'optimisation difficile. Editions Eyrolles.

[Dréo, 2004] Dreo Johann, « Adaptation de la méthode des colonies de fourmis pour l'optimisation en variables continues. Application en génie biomédical », Thèse de Doctorat de l'Université de Paris 12, soutenue en 13 décembre 2004.

[Dubuisson, 2006] Dubuisson Duplessis G. and Myceky P. « Ant Colony Optimization ». INSA de Rouen, Département Génie Mathématique, 2006.

[Duda, 1973] Duda R. O., & Hart P. E. (1973). Pattern classification and scene analysis. John Wiley & Sons.

[Dudot, 2005] A. Dutot, A. Cardon, D. Olivier, et F. Guinand, Competing ants for organization detection, application to dynamic distribution. Proc. of the ECCS 05, European conference on complex systems. 2005

[Emmanuel,2000] Emmanuel Tonye, Alain Akono, André NDI Nyoungui. Traitement des images de télédétection par l'exemple. Gordon and Breach Science Publisher 2000.

[Farhad,2010] Farhad Samadzadegan, Sara Saeedi, Hadiseh Hasani, 2010. Evaluating the potential of ant colony optimization in clustering of lidar data. 24. - 27. 1. 2010, Ostrava.

[Fekraoui, 2005] Fekraoui Farah. Classification des images satellitaires à haute résolution « Application à une image TM de LANDSAT-5 de la région d'Oran en Algérie. Thèse de Magister. CNTS, Décembre 2005

[Floyd, 1987] Floyd F., Jr. Sabins. Remote Sensing: Principles and Interpretation." 2nd Edition, Better World Books Ltd.

[Focus, 1996] Focus Dergisi *(*Le magazine Focus*)*, octobre 1996.

[Forgy, 1965] Forgy E. W. Cluster Analysis of Multivariate Data : Efficiency Versus Interpretability of Classifications, Biometrics, 21, pp. 768-780.

[Gambardella, 1997] L. Gambardella, M. Dorigo, HAS-SOP: An hybrid ant system for the sequential ordering problem, Technical Report (11), IDSIA, Lugano, CH. 1997

[Gambardella, 1999] L.M. Gambardella, E. Taillard, and M. Dorigo. Ant colonies for the quadratic assignment problem. *Journal of the Operational Research Society*, 50 :167–176, 1999.

[Gary, 1979] M. Garey and D. Johnson. Computers and Intractability, A Guide to the Theory of NP-Completeness. W. H. Freeman and Company, San Francisco, 1979.

[Georgé, 2004] J-P Georgé Résolution de problèmes par émergence, Etude d'un Environnement de Programmation Emergente , Thèse de l'Université Paul Sabatier, Toulouse, Juillet 2004.

[Germond, 1999] L. Germond Trois principes de coopération pour la segmentation en imagerie de résonance magnétiques. Thèse de doctorat de l'université Joseph Fourrier Grenoble I. 11 Octobre 1999.

[Glineur, 1997] Glineur FRANÇOIS. « Etude des méthodes de point intérieur appliquées à la programmation linéaire et à la programmation semidéfinie ». Travail de fin d'études (*Engineering thesis*), Faculté Polytechnique de Mons, Belgium, June 1997

[Glover, 1989] Glover C. & Greenberg H.J., « New Approaches for Heuristic Search : A

Bilateral Linkage with Artificial Intelligence », European Journal of Operational Research, Vol. 39, N°. 2, pp 119-130, 1989.

[Gold, 1989] Goldberg, D. E. (1989). Genetic Algorithms in Search, Optimization, and Machine Learning. Reading MA: Addison Wesley.

[Goss, 1989] Goss S., Aron S., Deneubourg. J-L , Pasteels J.M., Self organized shortcuts in the argentine ant, Naturwissenschaften, Vol. 76, 1989, pp. 579-581.

[Grumbach, 1997] A. Grumbach. A propos d'émergence. Emergence ou explication. Intellecta emrgence and explanation 1997/2 n°0984 185-194, 1997.

[Habib, 1998] M. Habib. Polycopié du cours d'algorithmique. DEA d'informatique, Université de Montpellier II, 1998.

[Handl, 2003] Handl J. Ant_based methods for tasks of clustering znd topograpgoc mapping:extensions, analysis and comparison with alternative methods. Master thesis Germany. November 2003.

[Handl, 2006] Handl, J., Knowles, J., Dorigo, M. Ant-based clustering and topographic mapping. Artificial Life 12 (2006) 35-61

[Harun, 2003] Harun Y (2003) . Le miracle de la fourmi. Editions Editions AL-MADINAH, 2003, pp. 25

[Helton, 2005] Helton.N, Benemar.A ,Helivo.A ,"Banks of automatic capacitors in electrical distribution systems a hybrid algorithm of control"Revista Control&Automaçao, Vol.16 no.1/Jan.,Fev.eMarço2005.

[Heppner, 1990] Heppner F. & Grenander U., "A stochastic nonlinear model for coordinated bird flocks", AAAS Publication Washington, DC, 1990.

[Holl, 1975] Holland, J. (1975). Adaptation in natural and artificial systems. University of Michigan Press, Ann Arbor.

[Jafar, 2010] O.A. Mohamed Jafar and R. Sivakumar. Ant-based Clustering Algorithms: A Brief Survey. International Journal of Computer Theory and Engineering, Vol. 2, No. 5, October, 2010 1793-8201

[Jain, 1988] JainA.and Dubes R. (1988). Algorithms for Clustering Data.Prentice Hall Advanced Reference Series.

[Jensen, 1983] Jensen, J.R, 1983. Biophysical Remote Sensing. Annals of the association of American Geographer. Vol.73, p111-132.

[Jodogn, 2001] Jodogn S. Optimisation par une colonie d'agents coopérants. Université de Liège, 2001.

[Kao,2006] Y. Kao and K. Cheng. Tatung. An ACO-Based Clustering Algorithm. University, Taipei, Taiwan. ANTS 2006, LNCS 4150, pp. 340–347, 2006. Springer-Verlag Berlin Heidelberg 2006

[Kenn, 1995] Kennedy, J., & Eberhart, R. (1995). Particle Swarm Optimization. In Proceedings of IEEE International Conference on Neural Network, vol. IV, pp. 1942–1948.

[Kévin, 2006] Kévin L. Les algorithmes métaheuristiques. Licence : Creative Commons. Date : 01 Juin 2006.

[Khedam,2008] Khedam R. (2008). Contribution au développement de méthodologies de fusion/classification contextuelles d'images satellitaires multi-sources. Application à la cartographie thématique du milieu urbain de la ville d'Alger. Thèse de doctorat de l'université des Sciences et de la Technologie Houari BOUMEDIENE USTHB. Discipline : Electronique.

[Khouadjia, 2008] Khouadjia M.R., Batouche M. & Chikhi S. "Hybridation co-évolutionnaire entre méta-heuristique d'optimisation: la colonie de fourmis Pachycondyla apicalis et l'optimisation par essaim particulaire. Maghrebian Conference On Information Technologies. 10th Conference on Software Engineering and Artificial Intelligence. April 28-30, 2008 Oran ,

Algeria.

[Kuntz, 1997] P. Kuntz, P. Layzell et Snyers. D. A colony of ant- like agents for partitioning in vlsi technology. In P. Husbands et I. Harvey, éditeurs, Proceedings of the Fourth European Conference on Artificial Life, pages 417-424. MIT Press, 1997.

[Labroche, 2003] N.Labroche. Modélisation du système de reconnaissance chimique des fourmis pour le problème de la classification non-supervisé: application à la mesure d'audience sur Internet. Thèse pour obtenir le grade de docteur de l'université de Tours, 4 décembre 2003.

[Langham, 1999] A.E. Langham, P.W. Grant. Using competing ant colonies to solve k-way partitioning problems with foraging and raiding strategies. in: D. Floreano et al. (Eds.), Advances in Artificial Life, Lecture Notes in Computer Science, 1674, Springer, 1999, pp. 621–625.

[Lenoir, 2009] Lenoir A et Monmarché N. Fourmis artificielles 1 : des bases de l'optimisation aux applications industrielles. Lavoisier 2009

[Liu, 68] G. L. Liu. Introduction to combinatorial Mathematics. McGraw Hill, 1968.

[Lumer, 1994] Lumer, E. and Faieta, B., 1994. Diversity and Adaptation in Populations of Clustering Ants. Proceedings of the third International Conference on Simulation of Adaptive Behaviour: From Animals to Animates. MITPress, 501508.

[MacQueen, 1967] MacQueen L. Some methods for classification and analysis of multivariate observations. Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, 1 :281{297, 1967.

[Maniezzo, 1994] V. Maniezzo, A. Colorni, M. Dorigo. The Ant System Applied to the Quadratic Assignment Problem. Technical Report. IRIDIA. (28), Université Libre de Bruxelles, Belgium, 1994.

[Martinoli, 1999] A. Martinoli, A. Ijspeert, et L. Gambardella. A Probabilistic Model for Understanding and Comparing Collective Aggregation Mechanisms In (Floreano et al., 1999), pages 575–584. 1999.

[Maulik, 2003] U. Maulik et S. Bandyopadhyay, Fuzzy partitioning using a real-coded variable-length genetic algorithm for pixel classification, GeoRS(41), No. 5, pp. 1075-1081. 2003.

[McQueen , 1967] J. McQueen. Some methods for classification and analysis of multivariate observations. Proceedings of the fifth Berkeley Symposium on Mathematical Statistics and Probability, pages 281–297, 1967.

[Melhuish, 1999] C. Melhuish. Exploiting Domain Physics : Using Stigmergy to Control Cluster Building with Real Robots . In (Floreano et al., 1999), pages 585–595. 1999

[Merkel, 2002] D. Merkle, M. Middendorf and H. Schmeck, Ant colony optimisation for resource constrained project scheduling, IEEE Transaction on Evolutionary Computation, 6(4): 333-346 August 2002.

[Mich, 1996] Michalewicz, Z. (1996). Genetic Algorithms + Data Structures = Evolution Programs. 3rd Edition, Springer Verlag, Berlin.

[Monga, 1987] Monga O., Wrobel B. Segmentation d'images: vers une méthodologie. Traitement du Signal, vol. 4, nº 3, pp 169-193, 1987.

[Monm, 1999] Monmarché N., Venturini G., Slimane M. AntClass : discovery of clusters in numeric data by an hybridization of an ant colony with the Kmeans algorithm. Internal Report No 213, E3i, January 1999.

[Monm, 2000a] Monmarché N. (2000). Algorithmes de fourmis artificielles : applications à la classification et à l'optimisation. Thèse de Doctorat de l'université de Tours. Discipline : Informatique. Université François Rabelais, Tours, France, 231 p.

[Monm, 2000b] N. Monmarché, G. Venturini, M. Slimane. Classification non supervisee par une population de fourmis artificielles. Actes Coll. Insectes Sociaux, 13 : 43-52 (2000).

[Monm, 2009a] Monmarché N., Guinand F. & Siarry P. (2009). Fourmis artificielles 1 : des bases de l'optimisation aux applications inductrielles . Lavoisier.

[Monm, 2009b] Monmarché N., Guinand F. & Siarry P. (2009). Fourmis artificielles 2 : nouvelles directions pour une intelligence collective. Lavoisier.

[Murthy, 1996] C.A. Murthy, N. Chowdhury, In Search of Optimal Clusters Using Genetic Algorithms, Pattern Recognition Letters, vol.17, No. 8, , pp. 825-832, 1996.

[Olivier, 2004] Olivier A. Exploitation d'un modèle numérique de terrain en matière de gestion de l'environnement. Alain Olivie Géomatique 2004.

[Ouadfel, 2005] Ouadfel S., M. Batouche. Des fourmis pour la segmentation des images. Journées d'étude en Informatique graphique JIG05, Biskra novembre 2005.

[Ouadfel, 2006] Ouadfel S. (2006). Contributions à la Segmentation d'images basées sur la résolution collective par colonies de fourmis artificielles, Thèse de doctorat de l'université de Batna. Discipline : Informatique.

[Papadimitriou, 1994] Papadimitriou, C. 1994. Computational Complexity. AddisonWesley.

[Parpinelli, 2002] Parpinelli R.S., Lopes H.S. and Freitas A.A. (2002). Data Mining : a Heuristic Approach, chapter An Ant Colony Algorithm for Classification Rule Discovery, pages 191–208. London : Idea Group Publishing.

[Piec, 1989] Pieczynski W. (1989)**.** Estimation of context in random fields. Journal of Applied Statistics, 16, 2, pp. 283-289.

[Pierrick, 2010] Pierrick B. Contributions en classification automatique : agrégation bayésienne de mélanges de lois et visualisation interactive. Thèse de docorat. Université de Nantes. Spécialité Informatique.

[Proc, 1998] Proctor, G., & Winter, C. (1998). Information Flocking: Data Visualisation in Virtual Worlds using Emergent Behaviours. Heudin J.-C., Ed., Proc. 1st Int. Conf. Virtual Worlds, VW, vol. 1434, p. 168–176, Springer-Verlag.

[Raghavan, 1979] V.V. Raghavan et K. Birchard. A clustering strategy based on a formalism of the reproductive process in natural systems. In Information Implications into the Eighties, Proceedings of the Second International Conference on Information Storage and Retrieval, pages 10–22. ACM, 1979

[Ricardo, 2007] Ricardo Poli, James Kennedy & Tim Blackwell, "Paricle Swarm Optimization", In Swarm Intelligence 1(1):33-57, 2007.

[Rich, 1993] Richards J. A. (1993). Remote Sensing Digital Image Analysis. An Introduction. 2nd Edition, Springer-Verlag, Berlin.

[Rich, 1998] Richards J. A., & Jia X. (1998)**.** Remote Sensing Digital Image Analysis. An Introduction. 3rd Edition, Springer-Verlag, Berlin.

[Rida, 2006] Rida Mohamed A. Proposition d'une méthodede classification dans un environnement de robotique collective. Thèse de magistère en informatique 2006. Université de Batna.

[Russel, 2001] Russel C. Eberhart, Yuhui Shi & James Kennedy, « Swarm Intelligence », The Morgan Kaufmann Series in Artificial Intelligence. Morgan Kaufmann, San Fransisco, CA, USA, 2001.

[Russel, 2010] Stuart Russel & Peter Norvig. « Intelligence artificielle, avec plus de 500 exercices ». Cet ouvrage est la traduction de Artificial Intelligence : A modern approch. «3ème édition, publié par Prentice Hall, Pearson Education France, Copyright © 2010.

[Saporta, 1990] G. Saporta, Probabilités, analyse des données et statistique , éditions Technip, Paris, 1990, ISBN 2-7108-0565-0.

[Schi, 1996a] Schistad Solberg A. H., Taxt T. & Jain A. K. (1996). A Markov random field model for classification of multisource satellite imagery. IEEE Transactions on Geoscience and Remote Sensing, vol. 34, no. 1, pp. 100-113.

[Scho, 2004] Steven Schockaert, Martine De Cock, Chris Cornelis, and Etienne E. Kerre. Fuzzy ant based clustering. In Proceedings of ANTS2004, LNCS, 2004.

[Semet, 2003] Y. Semet, E. Lutton and P.Collet. Ant Colony Optimization for e-learning: Oberserving the emergence of pedagogic suggestions. In IEEE Swarm Inteligence Symposium 2003, Indianapolus, Indiana, april 2003.

[Shelokar, 2004] P.S. Shelokar, V.K. Jayaraman and B.D. Kulkarni, «An ant colony approach for clustering». Analytica Chimica Acta, Vol. 509, Issue 2, 3 May 2004, pp. 187-195.

[Shelokar, 2004] P.S. Shelokar, V.K. Jayaraman and B.D. Kulkarni. An ant colony approach for clustering. Analytica Chimica Acta, Vol. 509, Issue 2, 3 May 2004, pp. 187-195.

[Smara, 1998] Smara Y. (1998)**.** Contribution au système d'analyse et traitement d'image satellitaires L.A.T.L.S. Evaluation de correspondance terrain – image classifiée et intégration d'image multisources optique et radar SAR. Thèse de doctorat d'Etat, USTHB, Alger, 321 p.

[Solnon, 2000] C. Solnon. Solving permutation constraint satisfaction problems with artificial ants. In Proceedings of ECAI'2000, IOS Press, Amsterdam, The Netherlands, pages 118–122, 2000.

[Stützle, 1999] T. Stuzle, H.H. Hoos. The max-min ant system and local search for Combinatorial Problems, in Meta-heuristics : advances and trends in local search paradigms for Optimization. by S. VOSS, I.H. OSMAN and C. ROUCAIROL, Kluwer Academic Publishers, Boston, 1999, p. 313-329.

[Stützle, 1999b]Stützle, T. and Dorigo, M. (1999b). ACO Algorithms for the Traveling Salesman Problem. InMiettinen, K.,Mäkelä, M., Neittaanmäki, P., and Periaux, J., editors, Evolutionary Algorithms in Engineering and Computer Science : Recent Advances in Genetic Algorithms, Evolution Strategies, Evolutionary Programming, Genetic Programming and Industrial Applications. John Wiley & Sons.

[Stützle, 2000] T. Stützle, H. H. Hoos, MAX-MIN ant system, Future Generation Computer Systems, Vol. 16 (2000)

[Taillard, 1997] Taillard, L., Gambardella, L. (1997). An Ant Approach for Structured Quadratic Assignment Problems. 2nd Metaheuristic International Conference (MIC-97), Sophia-Antipolis, France.

[Taillard, 1999] E. Taillard, Ant systems, Handbook of Applied Optimization. P. PARDALOS, M.G.C RESENDE.

[Taillard, 1998] TAILLARD É. D., Programmation à mémoire adaptative et algorithmes pseudogloutons: nouvelles perspectives pour les méta-heuristiques, thèse d'habilitation, Université de Versailles, France, 1998.

[Takah, 1995] K. Takahashi, H. Nakatani, K. Abe. Color image segmentation using ISODATA clustering method. in: Second Asian Conf. on Computer Vision, Singapore, vol. 1, 1995, 523–527.

[Talbi, 2002] Talbi E.G., « A Taxonomy of Hybrid Metaheuristics », Journal of Heuristics, Vol. 8, pp. 541-564, 2002.

[Tfaili, 2007] Walid TFAILI. Conception d'un algorithme de colonie de fourmis pour l'optimisation continue dynamique. Thèse de doctorat de l'université PARIS 12-VAL DE MARNE UFR de Sciences et Technologie. Spécialité : SCIENCES DE L'INGÉNIEUR.

[Theraulaz, 1997] G. Theraulaz, F. Spitz. Auto-organisation et comportement. Hermès, Paris,

1997.

[Topin, 1999] Topin X.,"Etude de l'auto-organisation par coopération appliquée à l'apprentissage comportemental de robots fourmis", DEA RCFR – Université Paul Sabatier, Toulouse Juin 1999

[Trejos, 2004] J. Trejos, E. Piza, A. Murillo and E. Pizza. Clustering by Ant Colony Optimization. *University of Costa Rica, San José, CostaRica.*2004, Volume 0, Part I, 25-32, DOI: 10.1007/978-3-642-17103-1_3

[Trejos, 2007] Trejos J., Piza E., Murillo A. et Villalobos M. Partitionnement par colonies de fourmis et essaims particulaires. *Université du Costa Rica, San José, CostaRica.* XIVes Rencontres de la Société francophone de classification SFC'2010. Paris 5,6 et septembre 2007.

[Tsai, 2004] C.F. Tsai, C.W. Tsai, H.C. Wu and T. Yang. Acodf: a novel data clustering approach for data mining in large databases. Journal of System Software, 2004, 73(1), pp. 133-145.

[Tseng, 2001] L.Y. Tseng, S.B. Yang, A genetic approach to the automatic clustering problem, Pattern Recognition, vol. 34, No. 2, pp. 415-424, 2001

[Ünsal, 1993] C. Ünsal. Self-organization in large populations of mobile robots. Master of Sciences in Electrical Engineering, May 1993, Blacksburg, Virginia. http://armyant.ee.vt.edu/unsalWWW/cemthesis.html.

[Van De Vijver, 1997] Van De Vijver.G, Emergence et explication, Intellectica, "Emergence and explanation", 1997/2 n°25, ISSN n°0984-0028 185-194, 1997.

[Varela, 1988] F. Varela Autonomie et connaissance : essai sur le vivant Editions du seuil– 1988.

[Wallach, 1986] Wallach Y. "Calculation and programs for power systems network". Prentice-hall, Englewood Cliffs, 1986.

[Wilson, 1985] Wilson.S, Knowledge Growth in an Artificial Animal, in Proceedings of the 1st International Conference on Genetic Algorithms and their Applications, J.J. Grefenstette Ed., 1985, pp16-23.

[Wilson, 1988] Wilson E., and Hölldobler B. Dense Heterarchy and mass communication as the basis of organization in ant colonies. Trend in Ecology and Evolution, 3 :65-68.

[Yann-Chang, 1996] Yann-Chang Hong-tzer Yang Ching-Lien Huang "Solving the Capacitor Placement Problem in Radial Distribution System Using tabou Search Approach "IEEE Transaction on Power Systems Vol,11 No4 November pp1868-1873. Nov 1996.

[Yucheng, 2006]Yucheng Kao and Kevin Cheng, 2006. An ACO-Based Clustering Algorithm. 5th International Workshop, ANTS 2006. Brussels, Belgium, September4–7, 2006 Proceedings.

Webographie

[AaronFoltz, 2012] URL :
https://github.com/AaronFoltz/Heuristic-Genetic-Algorithm-for-the-Traveling-Salesman-Problem
Ce site a été consulté entre les dates [01 janvier 2012, 13 janvier 2012]
[Krippendorff, 1997] K. Krippendorff. Dictionnary of Cybernetics. 27 janvier 1997. URL:
http://pespmc1.vub.ac.be/SELFORG.html
Ce site a été consulté entre les dates [01 janvier 2011, 13 janvier 2012]
[MLRUCI, 2012] Machine Learning Repository at the University of California, Irvine.
URL :
http://archive.ics.uci.edu/ml/
[Wiki, 2012a] Site wikipedia. URL :
fr.wikipedia.org/wiki/Métaheuristique.

Ces deux derniers sites ont été consultés entre les dates [01 janvier 2012, 13 janvier 2012]

Glossaire

Glossaire

Biomimétique La biomimétique est une technique qui étudie et s'inspire de la nature pour résoudre des problèmes quotidiens. C'est une nouvelle discipline basée non pas sur ce qui peut être extrait de la nature mais sur ce qui peut être appris d'elle. Par exemple, l'araignée est capable de tisser une soie cinq fois plus résistante que l'acier, et arrive à cette « prouesse » sans température élevée (consommatrice d'énergie) et sans émettre de dioxine.

Carte thématique Une carte thématique illustre la répartition spatiale des données relatives à un thème ou plus pour les régions géographiques normalisées. La carte peut être de nature qualitative (p. ex. principaux types de fermes) ou quantitative (p. ex. variation en pourcentage de la population)

Classement Classification en anglais. Processus d'affectation d'un objet à une classe

Classification Clustering en anglais. Processus de découverte de classes.

Classes de complexité Les classes de complexité ont été introduites pour les problèmes de décision, c'est-à-dire les problèmes posant une question dont la réponse est « oui » ou « non ». Pour ces problèmes, on définit notamment les deux classes : P et NP :

- La classe P contient l'ensemble des problèmes polynomiaux, i.e., pouvant être résolus par un algorithme de complexité polynomiale. Cette classe caractérise l'ensemble des problèmes que l'on peut résoudre « efficacement ».
- La classe NP contient l'ensemble des problèmes polynomiaux non déterministes, i.e., pouvant être résolus par un algorithme de complexité polynomiale pour une machine non déterministe (que l'on peut voir comme une machine capable d'exécuter en parallèle un nombre fini d'alternatives). Intuitivement, cela signifie que la résolution des problèmes de NP peut nécessiter l'examen d'un grand nombre (éventuellement exponentiel) de cas, mais que l'examen de chaque cas doit pouvoir être fait en un temps polynomial

Complexité d'un problème Est une estimation du nombre d'instructions à exécuter pour résoudre les instances de ce problème, cette estimation étant un ordre de grandeur par rapport à la taille de l'instance. Il s'agit là d'une estimation dans le pire des cas dans le sens où la complexité d'un problème est définie en considérant son instance la plus difficile. Les travaux théoriques dans ce domaine ont permis d'identifier différentes classes de problèmes en fonction de la complexité de leur résolution [Papadimitriou, 1994]. En effet, il existe un très grand nombre de classes différentes, et on se limitera ici à une présentation succincte nous permettant de caractériser formellement la notion de problème combinatoire.

Contrôle décentralisé chaque fourmi décide de manière autonome en fonction de son environnement proche

Couvain Œuf d'insecte

Diversification/ Intensification Par intensification, on entend l'exploitation de l'information rassemblée par le système à un moment donné. La diversification est au contraire l'exploration de régions de l'espace de recherche imparfaitement prises en compte. Il va s'agir de choisir où et quand « injecter de l'aléatoire » dans le système (diversification) et/ou améliorer une solution (intensification).

Ethologie Science qui étudie, et décrit le comportement des animaux dans leur environnement.

Empirique Qui procède plus par tâtonnement, par sentiment que par réelle connaissance scientifique. Ainsi, empiriquement l'homme a eu le sentiment que la Terre était un disque, et que le soleil tournait autour d'elle. Il en a déduit la suprématie de la Terre et des hommes, faits par Dieu à son image. Galilée a mis en évidence le système solaire, par une approche scientifique.

Essaim Colonie d'abeilles sortant de la ruche mère pour fonder une nouvelle ruche. Troupe nombreuse

Essaims particulaires L'optimisation par essaims particulaires (OEP ou PSO en anglais) est une métaheuristique d'optimisation, inventée par Russel Eberhart (ingénieur en électricité) et James Kennedy (socio-psychologue) en 1995. Cet algorithme s'inspire à l'origine du monde du vivant. Il s'appuie notamment sur un modèle développé par Craig Reynolds à la fin des années 1980, permettant de simuler le déplacement d'un groupe d'oiseaux. Cette méthode d'optimisation se base sur la collaboration des individus entre eux. Elle a d'ailleurs des similarités avec les algorithmes de colonies de fourmis, qui s'appuient eux aussi sur le concept d'auto-organisation. Cette idée veut qu'un groupe d'individus peu intelligents puisse posséder une organisation globale complexe.

Heuristique Une heuristique est l'utilisation de règles empiriques. C'est aussi une technique qui consiste à apprendre petit à petit, en tenant compte de ce que l'on a fait précédemment pour tendre vers la solution d'un problème. L'heuristique ne garantit pas du tout qu'on arrive à une solution quelconque en un temps fini. Opposé à algorithmique.

Histogramme Est un graphique statistique permettant de représenter la distribution des intensités des pixels d'une image, c'est-à-dire le nombre de pixels pour chaque intensité lumineuse. Par convention un histogramme représente le niveau d'intensité en abscisse en allant plus foncé (à gauche) au plus clair (à droite). Par exemple, un histogramme d'une image en 256 niveaux de gris sera représenté par un graphique possédant 256 valeurs en abscisse, et le nombre de pixels de l'image en ordonnées.

Myrmécologie Etude du comportement naturel des fourmis.

Niveau de gris Est la valeur de l'intensité lumineuse en un point. La couleur du pixel peut prendre des valeurs allant du noir au blanc en passant par un nombre fini de niveaux intermédiaires. Le nombre de niveaux de gris dépend du nombre de bits utilisés pour décrire la « couleur » de chaque pixel de l'image. Plus

ce nombre est important plus les niveaux de gris possibles sont nombreux

Phénotype Aspect observable de l'individu, conditionné par son génotype et le milieu environnant.

Phéromone Les phéromones sont des hormones émises par la plupart des animaux et qui agissent comme des messagers sur des individus de la même espèce. Extrêmement actives elles agissent en quantités infinitésimales, si bien qu'elles peuvent être détectées, même transportées à plusieurs kilomètres. Chez les mammifères et les reptiles, les phéromones sont détectées par l'organe voméro-nasal, tandis que les insectes utilisent généralement leurs antennes.

Pixel Le pixel, souvent abrégé px, est l'unité de base permettant de mesurer la définition d'une image numérique matricielle. Son nom provient de la locution anglaise « picture element », qui signifie « élément d'image ».

Satellites Un satellite de télédétection est un satellite artificiel dont l'objectif principal est l'observation vers le bas, c'est-à-dire vers l'astre autour duquel il orbite (le plus souvent la Terre) à des fins civiles.

Stigmergie C'est un mode de communication indirecte via la modification dynamique de l'environnement. Il a été baptisé par le biologiste Pierre-Paul Grassé, en 1959.

Télédétection La télédétection désigne, dans son acception la plus large, la mesure ou l'acquisition d'informations sur un objet ou un phénomène, par l'intermédiaire d'un instrument de mesure n'ayant pas de contact avec l'objet étudié. C'est l'utilisation à distance de n'importe quel type d'instrument (par exemple, d'un avion, d'un engin spatial, d'un satellite ou encore d'un bateau) permettant l'acquisition d'informations sur l'environnement.