Modèle bio-inspiré pour le clustering de graphes : application à la fouille de données et à la distribution de simulations

Nesrine Masmoudi

Normandie Université

THESE

Pour obtenir le diplôme de doctorat

Spécialité Informatique

Préparée au sein de « Université du Havre»

En partenariat international avec « Université de Sfax» « Tunisie»

Modèle bio-inspiré pour le clustering de graphes, applications à la fouille de données et à la distribution de simulations

Présentée et soutenue par
Nesrine MASMOUDI épouse HENTATI

Thèse soutenue publiquement le 6 Janvier 2017 devant le jury composé de		
Cyrille BERTELLE	Professeur, Université du Havre	Directeur
Hanene AZZAG	Maître de conférences HDR, Université Paris Nord	Encadreur
Mustapha LEBBAH	Maître de conférences HDR, Université Paris Nord	Encadreur
Maher BEN JEMAA	Professeur, Université de Sfax	Directeur
Cyril DE RUNZ	Maître de conférences HDR, Université de Reims Champagne-Ardenne	Rapporteur
Faiez GARGOURI	Professeur, Université de Sfax	Président
Khaled GEDIRA	Professeur, Université de Tunis	Rapporteur

Thèse dirigée par Cyrille BERTELLE et Maher BEN JEMAA, laboratoires LITIS et ReDCAD

UNIVERSITÉ LE HAVRE

SPM ECOLE DOCTORALE

litis

TABLE DES MATIÈRES

Introduction 8

1 Fourmis réelles comme source d'inspiration **13**

1.1 Introduction 14

1.2 La bio-inspiration et l'intelligence en essaim 15

1.3 Intelligence en essaim, auto-organisation et systèmes complexes . 17

1.3.1 L'auto-organisation : concept fondateur de l'intelligence en essaim 17

1.3.2 Les systèmes complexes pour l'intelligence en essaim . . 20

1.4 Champs de l'intelligence en essaim 22

1.4.1 L'intelligence collective des lymphocytes : système immunitaire 22

1.4.2 Robotique en essaim 22

1.4.3 L'intelligence artificielle distribuée et systèmes multi-agents 25

1.5 Propriétés biologiques des fourmis réelles et application en informatique 27

1.5.1 La division du travail 28

1.5.2 La communication chez les fourmis 28

1.5.3 La construction du nid chez les fourmis 30

1.5.4 Le rassemblement d'objets 31

1.5.5 Auto-assemblage chez les fourmis réelles 32

1.5.6 La stratégie de prédation 33

1.5.7 Reconnaissance chimique des fourmis réelles 34

1.6 Conclusion 36

2 Classification non supervisée **37**

2.1 Introduction 39

2.2 La classification 40

2.2.1 Contexte de la classification 40
2.2.2 Définition du problème 41
2.2.3 Taxonomie des méthodes de classification ou clustering . 41
2.2.4 Métrique et distance 42
2.2.5 Types de données en classification 43
2.3 Méthodes non hiérarchiques 43
2.3.1 Méthode K-Means 44
2.4 Méthodes hiérarchiques 46
2.4.1 Classification Hiérarchique Ascendante 47
2.4.2 Classification Hiérarchique Descendante 50
2.5 Classification incrémentale et flux de données 50
2.5.1 L'algorithme BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) 50
2.5.2 L'algorithme CURE (Clustering Using REpresentatives) . 51
2.5.3 L'algorithme COBWEB 52
2.5.4 L'algorithme CluStream 53
2.5.5 L'algorithme DenStream 53
2.5.6 L'algorithme D-Stream 53
2.5.7 L'algorithme ClusTree 54
2.6 Les méthodes à base de grille 54
2.6.1 Les cartes auto-organisatrices (Self Organizing Maps) de Kohonen 54
2.7 Approches biomimétiques pour la classification 55
2.7.1 Fourmis artificielles 56
2.7.2 Autres approches biomimétiques : les algorithmes d'optimisation 56
2.7.3 Les algorithmes de classification automatique par fourmis artificielles 60
2.8 Méthodes de clusetring basées sur les graphes 72
2.8.1 Les graphes de voisinage et les méthodes de visualisation des graphes 73
2.8.2 CHAMELEON 81
2.8.3 Propagation d'Affinité 82
2.8.4 NG + CHL : Méthode à deux niveaux basée sur les graphes 83
2.8.5 Méthodes biomimétiques et les graphes 84
2.9 Conclusion 86

3 L'algorithme initial CL-Ant 89
3.1 Introduction 90
3.2 Le principe général de l'algorithme CL-Ant 91
3.2.1 Fourmis artificielles : principes 93

3.2.2 Description de l'algorithme 95
3.3 Expérimentations et résultats 104
3.3.1 Mesures d'évaluation 104
3.3.2 Les jeux de données et la mesure de similarité 106
3.3.3 Résultats 110
3.4 Visualisation des résultats 113
3.5 Conclusion 116

4 Extension de l'algorithme CL-Ant : version incrémentale 119
4.1 Introduction 120
4.2 Rappel des principes généraux de notre modèle de fourmis 120
4.3 Description de l'algorithme CL-AntInc 121
4.3.1 Principes 121
4.3.2 Choix de l'échantillon initial 123
4.3.3 Les phases de classification 124
4.4 Etude expérimentale et validation 125
4.4.1 Jeux de données utilisés 125
4.4.2 Résultats 128
4.5 Visualisation 132
4.6 Conclusion 133

5 Étude comparative 135
5.1 Introduction 136
5.2 Algorithmes étudiés : comparaison avec des méthodes traditionnelles 136
5.2.1 Expérimentations 137
5.2.2 Comparaison avec les méthodes classiques 141
5.2.3 Temps d'exécution 141
5.3 Comparaison avec des méthodes de flux 143
5.3.1 Expérimentations 143
5.3.2 Étude comparative par les données numériques 143
5.3.3 Temps d'exécution 145
5.4 Étude de la Complexité 146
5.5 Propriétés de l'algorithme 148
5.6 Conclusion 149

Conclusion générale 150

A Publications scientifiques 155

Bibliographie 157

TABLE DES FIGURES

1.1 Exemples de comportements collectifs dans les sociétés animales et humaines [Garnier 2008]. 17
1.2 Le robot nommé «Robobee» : leurres intelligents [Michelsen 1989]. 24
1.3 (a) Le robot Genghis développé par Rodney Brooks [Brooks 1989]. (b) Le robot Sahabot développé par Lambrinos et al. [Lambrinos 2000] pour tester les hypothèses sur la navigation de la fourmi Cataglyphis. (c) William Gray Walter et son robot ELSIE rentrant à son poste de rechargement [Holland 1992]. (d) Le robot Insbot développé dans le cadre du projet européen Leurre pour interagir avec un groupe de blatte Periplaneta americana [Halloy 2007]. 26
1.4 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 [Khedam 2008]. 32

2.1 Hiérarchie des méthodes de classification selon Jain et Dubes [Jain 1988]. 42
2.2 Fonctionnement générique de l'algorithme K-Means. 45
2.3 Un dendogramme (ou «arbre») [Celeux 1989]. 47
2.4 Vue générale de l'algorithme D-Stream. 54
2.5 Fonctionnement de l'algorithme API : Les fourmis (cercles pleins) explorent des sites de recherche (petits carrés) dans un périmètre (grand cercle) autour du nid [Dréo 2004]. 60
2.6 Grille utilisée dans E.Lumer et B.Faieta [Monmarché 2000] : La fourmi est représentée par $\times$ et son périmètre de détection par un trait épais. Les objets sont représentés par des carrés dont l'intérieur («invisible» pour la fourmi) représente la classe d'origine. . . 62
2.7 AntClass permet la construction de tas d'objets sur la grille [Monmarché 2000]. 64

2.8 Construction de l'arbre par des fourmis : principe général. Les fourmis qui sont en déplacement sont représentées en gris et les fourmis connectées en blanc [Azzag 2007]. 66
2.9 Exemple du fonctionnement de l'algorithme TTA [Wong 2007b]. . 68
2.10 Exemple d'un graphe de voisinage. 74
2.11 Diagramme de Voronoi - a) Nuage de sites V dans un espace F de dimension 2. b) Diagramme de Voronoi correspondant à l'ensemble de sites V dans F. c) Triangulation de Delaunay ou dual du diagramme de Voronoi de la figure 2.11 b). 75
2.12 Cercle circonscrit aux 3 points - a) Triangulation de Delaunay. b) Un point appartient à la surface décrite par le cercle, ce n'est pas une triangulation. 77
2.13 Graphe de Gabriel de l'ensemble de points P. 77
2.14 Définition des Voisins Relatifs - a) L'intersection des hypershères H(v_i, v_j) et H(v_j, v_i) est vide. Les sommets v_i et v_j sont voisins. Cette relation est concrétisée, sur la figure, par l'existence d'un lien en trait plein entre v_i et v_j . b) Le sommet $v_k \in$ à l'intersection des hypersphères de centre v_i et v_j et de rayon |v_i - v_j|. Les points v_i et v_j ne sont donc pas voisins. 78
2.15 Processus général d'une exécution de l'algorithme CHAMELEON. 82
2.16 Exemple d'une exécution de l'algorithme Propagation d'Affinité [Frey]. Plus un point est rouge, plus il est candidat pour être un représentant. La couleur des arêtes est proportionnelle à la force des messages transmis. 82
2.17 Quelques étapes de l'apprentissage par Neural Gas + CHL. Les données sont dans la zone bleue, les prototypes sont en vert. Les connexions pertinentes sont créées au fur et à mesure. A la fin de l'apprentissage les régions de Voronoi déterminent quel prototype sera le plus représentatif de chaque donnée. 83

3.1 Construction du graphe par des fourmis : principe général. Les fourmis qui sont en déplacement sont représentées en gris et les fourmis affectées sont hachurées. Les autres fourmis ne sont pas encore testées. 93
3.2 Décision d'une fourmi de quitter son cluster d'appartenance. Une fourmi f_i en déplacement est en gris. 95
3.3 Règles de comportement de f_i si elle est affectée au nid/cluster voisin $CL_{j'}$ (voir l'algorithme 2). 96
3.4 Le graphe résultat de l'étape d'initialisation en utilisant l'algorithme K-Means : CL-Ant$_1$() 99
3.5 La première étape de clustering : CL$-Ant_2()$ 99

3.6 Les résultats de l'indice de pureté obtenu avec CL-Ant (Test 1, Test 2, Test 3) sur les bases de données numériques après 1000 itérations. 111
3.7 Les résultats de l'indice de pureté obtenu avec CL-Ant (Test 1, Test 2, Test 3) sur les bases de données binaires après 1000 itérations. . 111
3.8 Les résultats de l'indice de pureté obtenu avec CL-Ant (Test 1, Test 2, Test 3) en fonction du nombre d'itérations. 112
3.9 Exemple de graphe de proximité après l'algorithme CL-Ant pour la base de données Glass avec k = 14. 115
3.10 Exemple de graphe de proximité obtenu par l'algorithme CL-Ant pour la base de données Iris avec k = 10. 115
3.11 Exemple de graphe de proximité obtenu par l'algorithme CL-Ant pour la base de données Pima avec k = 9. 115
3.12 Exemple de graphe de proximité obtenu par l'algorithme CL-Ant pour la base de données breast cancer avec k = 4, α = 0.02 et γ = 0.001. 116
3.13 Exemple de graphe de proximité obtenu par l'algorithme CL-Ant pour la base de données primary Tumor avec k = 23, α = 0.02 et γ = 0.001. 116

4.1 Étape incrémentale : découper l'ensemble de données (pour simuler le flux) en N_B blocs. 121
4.2 Principe général de l'algorithme CL-AntInc. 124
4.3 Les résultats de l'indice de pureté obtenu avec CL-AntInc (Test 1, Test 2, Test 3) sur les bases de données numériques après 1000 itérations. 128
4.4 Les résultats de l'indice de Rand obtenu avec CL-AntInc (Test 1, Test 2, Test 3) sur les bases de données numériques après 1000 itérations. 129
4.5 Évolution de l'indice de pureté obtenu avec CL-AntInc (Test 2) après l'intégration de N_B blocs appliquée aux différents bases de données numériques. 129
4.6 (a) : Le pourcentage de fréquence de sortie des fourmis. (b) : L'évolution de l'indice de pureté obtenu par CL-AntInc (Test 1) après l'intégration de N_B blocs appliqué à différents bases de données. . 130
4.7 Les résultats de l'indice de pureté obtenu avec l'algorithme CL-AntInc (Test 1, Test 2, Test 3) sur les bases de données binaires après 1000 itérations. 131
4.8 Évolution du graphe obtenu à partir du l'algorithme CL-AntInc (Test 1) appliqué à la base de données binaire Car Evaluation avec K = 3. 132

4.9 Évolution du graphe obtenu à partir du l'algorithme CL-AntInc (Test 1) appliqué à la base de données numérique Statlog(Heart) avec K = 6. 133
4.10 Évolution du graphe obtenu à partir du l'algorithme CL-AntInc (Test 3) appliqué à la base de données numérique SpamBase avec K = 4. 133

5.1 Les résultats de l'indice de pureté obtenus avec CL-Ant (Test 3 : α = 0.03 et γ = 0.001), K-Means et CAH sur les bases de données numériques après 1000 itérations. 137
5.2 Les résultats de l'indice de pureté obtenus avec CL-Ant (Test 1, Test 2, Test 3), K-Means et CAH sur les bases de données numériques après 1000 itérations. 138
5.3 Les résultats de l'indice de Rand obtenus avec CL-Ant (Test 3 : α = 0.03 et γ = 0.001), K-Means et CAH sur les bases de données numériques après 1000 itérations. 139
5.4 Les résultats de l'indice de Rand obtenus avec CL-Ant (Test 1, Test 2, Test 3), K-Means et CAH sur les bases de données numériques après 1000 itérations. 139
5.5 Les résultats de l'indice NMI obtenus avec CL-Ant (Test 3 : α = 0.03 et γ = 0.001), K-Means et CAH sur les bases de données numériques après 1000 itérations. 140
5.6 Les résultats de l'indice NMI obtenus avec CL-Ant (Test 1, Test 2, Test 3), K-Means et CAH sur les bases de données numériques après 1000 itérations. 140
5.7 Le temps d'exécution en millisecondes obtenus avec CL-Ant (Test 1, Test 2, Test 3) sur des base de données numériques après 1000 itérations. L'algorithme a été programmé en Java et les tests ont été effectués sur une machine avec un processeur Pentium 52.6 GHz et 512 Mo de RAM. 142
5.8 Le temps d'exécution en millisecondes obtenus avec CL-Ant (Test 1, Test 2, Test 3) sur des bases de données binaires après 1000 iterations. L'algorithme a été programmé en Java et les tests ont été effectués sur une machine avec un processeur Pentium 52.6 GHz et 512 Mo de RAM. 142
5.9 Les résultats de l'indice NMI obtenus avec CL-AntInc (Test 1, Test 2, Test 3), DenStream, CluStream et D-Stream sur les bases de données numériques après 1000 itérations. 143
5.10 Les résultats de l'indice de pureté obtenus avec CL-AntInc (Test 2), DenStream, CluStream, D-Stream et ClusTree sur les bases de données binaires après 1000 itérations. 145

5.11 Le temps de calcul total en secondes obtenu par l'algorithme CL-AntInc pour les bases de données binaires (P4, 6GB RAM, avec Java). 145
5.12 Le temps d'exécution comparative en millisecondes entre l'algorithme CL-AntInc (Test 1, Test 2, Test 3) et les méthodes incrémentaux tels que DenStream, CluStream et D-Stream pour les ensembles de données. 146

LISTE DES TABLEAUX

2.1 Liste des algorithmes d'optimisation par colonies de fourmis. [Boulmerka 2009] 59

3.1 Jeux de données utilisées dans l'évaluation. 107
3.2 Paramètres testés pour CL-Ant. 108
3.3 Tableau des variables catégorielles codées en additif ; on remarque que dans les trois cas la médiane représente l'une des modalités de la variable. 108
3.4 Exemple : la variable taille codée en codage additif. 109
3.5 Table de contingence. a et d représentent le nombre de fois que les deux individus choisissent la même modalité «1» ou «0». b et c représentent le nombre de fois que le premier individu (deuxième individu) choisit la modalité «1» et le deuxième individu (le premier individu) choisit la modalité «0». 110

4.1 Description des bases de données utilisées par l'algorithme CL-AntInc (Machine Learning Repository). 126
4.2 Les bases de données numériques utilisées dans nos tests par l'algorithme CL-AntInc, où B_i est la taille différente des blocs choisie d'une manière aléatoire. 126
4.3 La taille des différents blocs obtenus par l'algorithme CL-AntInc sur des données binaires. 127

5.1 Les indices de Pureté et de Rand entre CL-AntInc et les différents algorithmes incrémentaux (DenStream, CluStream, D-Stream) ; Le poids sur l'arête est augmenté par la valeur α qui représente le taux de phéromone, tous les arêtes sont diminués par la valeur γ corresponds aux clusters de haute dissimilarité. 144
5.2 Calcul de complexité des algorithmes de classification 147

LISTE DES ALGORITHMES

1 Principe général de l'algorithme par des fourmis artificielles. . . . 97
2 Vue détaillée de l'algorithme CL-Ant. 98
3 Vue générale de l'algorithme non suppervisé de construction non hiérarchique du graphes de voisinage. 100
4 Une vue génnérique de l'algorithme de classification incrémentale CL-AntInc. 123

CITATION

S'il n'y avait pas d'hiver,
le printemps ne serait pas si agréable,
si nous ne goûtions pas à l'adversité,
la réussite ne serait pas tant appréciée.
(**Anne BRADSTREET**)

REMERCIEMENTS

Par ces quelques lignes, je tiens à remercier toutes les personnes qui ont participé de près ou de loin au bon déroulement de cette thèse, en espérant n'avoir oublié personne ...

Je voudrais tout d'abord remercier grandement mes encadrants Hanane AZZAG et Mustapha LEBBAH pour leur constante disponibilité et leurs précieux conseils avisés et leur encadrement régulier, qui ont permis à ce travail de voir le jour. Je suis ravi d'avoir travaillé avec eux car outre leur appui scientifique, ils ont toujours été présents pour me soutenir et me conseiller durant ces années. Je suis extrêmement sensible à leurs qualités humaines et à leur sens de l'écoute et de la compréhension. J'ai beaucoup apprécié travailler à vos côtés tant sur le plan scientifique que sur le plan humain. Je garde toujours beaucoup de plaisir à discuter avec vous et à bénéficier de vos conseils.

Je tiens à remercier sincèrement mon directeur de thèse français Pr. Cyrille BERTELLE, qui a dirigé mon travail : ses conseils, pour ses qualités scientifiques et ses commentaires précieux m'ont permis de surmonter mes difficultés et de progresser dans mes études. Et aussi pour le projet qu'il m'a permis de réaliser au sein de son équipe et la confiance dont il a fait preuve à mon égard. Je tiens à exprimer ma totale reconnaissance pour sa rigueur et ses encouragements tout au long de ce travail. Je garderai dans mon coeur votre générosité, votre compréhension et votre efficacité. Pour tout ce que vous m'avez donné, je vous remercie très sincèrement.

Je tiens à exprimer toute ma reconnaissance à mon directeur de thèse tunisien Pr. Maher BEN JEMAA pour son encadrement, ses nombreux conseils et son soutien constant tout au long de ma thèse. Vos remarques pertinentes et vos conseils précieux m'ont beaucoup aidé durant ces années.

Les Professeurs Cyril DE RUNZ et Khaled GEDUIRA pour avoir accepter d'être les rapporteurs de cette thèse. Ils ont également contribué par leurs nombreuses remarques et suggestions à améliorer la qualité de cette thèse, et je leur en suis très reconnaissante.

Mes sincères remerciements et ma gratitude au professeur Faiez GARGOURI d'avoir accepté de juger ce travail et d'en présider le jury de soutenance ; Que vous soyez assuré de mon entière reconnaissance. Je lui en remercie profondément.

Ma gratitude, mon profond respect et mes remerciements vont à tous les membres du jury : rapporteurs, examinateurs et encadreurs, pour leur attention consacrée à l'égard de mon travail.

Je remercie tous les chercheurs, enseignants et membres du personnel des laboratoires LITIS, LIPN et ReDCAD pour leur amitié et leur aide pendant ces années de thèse. Je tiens à témoigner tout particulièrement ma sympathie et ma reconnaissance à ces équipes. De plus, grâce à eux, j'aurai pu découvrir quelques pays.

RÉSUMÉ

Dans ce travail de thèse, nous présentons une méthode originale s'inspirant des comportements des fourmis réelles pour la résolution de problème de classification non supervisée non hiérarchique. Cette approche créée dynamiquement des groupes de données. Elle est basée sur le concept des fourmis artificielles qui se déplacent en même temps de manière complexe avec les règles de localisation simples. Chaque fourmi représente une donnée dans l'algorithme. Les mouvements des fourmis visent à créer des groupes homogènes de données qui évoluent ensemble dans une structure de graphe.

Nous proposons également une méthode de construction incrémentale de graphes de voisinage par des fourmis artificielles. Nous proposons deux méthodes qui se dérivent parmi les algorithmes biomimétiques. Ces méthodes sont hybrides dans le sens où la recherche du nombre de classes, de départ, est effectuée par l'algorithme de classification K-Means, qui est utilisé pour initialiser la première partition et la structure de graphe.

Mots clés
- Data Mining ;
- Classification ;
- Algorithmes biomimétiques ;
- Intelligence en essaim ;
- Fourmis artificielles.

ABSTRACT

In this work, we present a novel method based on behavior of real ants for solving unsupervised non-hierarchical classification problem. This approach dynamically creates data groups. It is based on the concept of artificial ants moving complexly at the same time with simple location rules. Each ant represents a data in the algorithm. The movements of ants aim to create homogenous data groups that evolve together in a graph structure. We also propose a method of incremental building neighborhood graphs by artificial ants. We propose two approaches that are derived among biomimetic algorithms, they are hybrid in the sense that the search for the number of classes starting, which are performed by the classical algorithm K-Means classification, it is used to initialize the first partition and the graph structure.

Keywords
– Data Mining ;
– Clustering ;
– Swarm intelligence ;
– Biomimetic algorithms ;
– Artificial ants.

INTRODUCTION

Nous vivons dans un monde chargé de données. Ce volume de données présent sur internet est en augmentation extrêmement rapide, il est donc nécessaire de penser à concevoir des méthodes efficaces permettant d'extraire, de classer, de manipuler et de mettre en forme les informations qu'elles peuvent contenir à partir des nouveaux algorithmes performants pour traiter ce nombre de données. Avec l'augmentation des capacités de traitement d'information, qui ont pour objectif de rechercher des similarités ou des relations de dépendance entre des ensembles d'objets. Elles sont pour l'essentiel, issues des recherches en statistique (analyse des données) et en intelligence artificielle (méthode d'apprentissage). D'une manière générale, l'objectif de ces méthodes est de réduire la dimension des données en fournissant une vue synthétique qui permet de déceler des groupes d'objets qui se ressemblent (classification) ou des objets dépendants.
Dès lors, plusieurs outils de décision, certains nouveaux, d'autres existants depuis longtemps, ont été regroupé sous le terme **«Data Mining»** permettant d'extraire de l'information pertinente à partir de volumes importants de données afin de prendre les bonnes décisions.

Dans cette étude, nous allons nous intéresser aux modèles qui copient les comportements des insectes sociaux, c'est à dire l'intelligence collective ou l'intelligence en essaim qu'il est possible d'observer chez les abeilles, les guêpes, certaines espèces de poissons, les termites... ou plus particulièrement chez la fourmi, qui sera la principale espèce étudiée. La classification, qui est l'une des tâches de Data Mining, fait partie des problèmes pour lesquels les fourmis suggèrent des heuristiques très intéressantes pour les informaticiens. Ces algorithmes sont des méthodes de résolution qui se basent sur une branche de l'intelligence artificielle et la modélisation de systèmes complexes naturels comme source d'inspiration.

Sujet de la thèse

Nous présentons dans cette thèse un nouvel algorithme de classification non supervisée et pour le partitionnement de graphes. Il utilise le principe de la construction d'odeur coloniale observée chez des fourmis réelles qui construisent des groupes de fourmis portant la même odeur coloniale en se déplacant d'un nid à un autre puis successivement affectées aux fourmis déjà assignées dans leur nids. Chaque fourmi artificielle représente une donnée à classer. Ces fourmis vont ensuite construire de manière similaire un graphe en appliquant certaines règles comportementales. Les déplacements et la construction d'une odeur coloniale des fourmis qui appartiennent au même nid sur un graphe dépendent de la similarité entre les données. Nous avons comparé cet algorithme à une méthode de partitionnement (K-Means) et à une méthode hiérarchique (classification ascendante hiérarchique). Nous avons ensuite proposé une extension de construction incrémentale d'un graphe de voisinage par des fourmis artificielles à partir d'un volume de données. Nous consacrons également une comparaison de notre approche vis à vis des méthodes de classification incrémentale (i.e. DenStream, CluStream, D-Stream et ClusTree). Nous avons testé les deux méthodes proposées sur des données numériques et binaires. Nous mesurons les temps d'exécution et la complexité des algorithmes, nous présentons, par la suite, l'aspect visuel des graphes construits avec nos méthodes non hiérarchiques.

Les objectifs que nous nous sommes amenés à suivre sont donc les suivants :

- L'usage de l'intelligence en essaim pour l'analyse dynamique des données à partir des graphes dynamiques,
- Définir un nouvel algorithme bio-inspiré qui crée dynamiquement des groupes de données. Cet algorithme est basé sur le concept de fourmis artificielles qui se déplacent de manière complexe avec des règles de localisation simples. Chaque donnée représente une fourmi. Les mouvements de fourmis visent à créer des groupes de données homogènes qui évoluent ensemble dans une structure de graphe,
- On suggère également une extension de cet algorithme appelé CL-AntInc pour traiter des données d'une manière incrémentale,
- Nous fournissons une comparaison analytique détaillée de nos résultats avec ceux obtenus par des méthodes classiques de classification non supervisées (hiérarchiques et de type partitionnement) et avec différentes méthodes de classification incrémentale ;
- Nous présentons l'aspect visuel des graphes construits avec les méthodes proposées sur un ensemble de bases de données à l'aide de la plateforme de visualisation des graphes «Tulip».

Organisation du document

Cette étude comporte 5 chapitres :

• Le premier chapitre sera dédié à détailler la notion de l'intelligence en essaim, ses caractéristiques et l'utilisation de cette notion inspirée du comportement des animaux dans le domaine informatique et l'intelligence artificielle (classification, robotique, etc).

• Le deuxième chapitre intitulé «Classification non supervisée» dresse un panorama des différentes approches proposées pour regrouper des données. Nous abordons quelques méthodes de la classification classique et aux principales techniques de classification incrémentales. Nous décrivons ensuite les approches biomimétiques qui se trouvent être une riche source d'inspiration pour la classification, en portant un intérêt plus marqué pour les méthodes inspirées du comportement des fourmis réelles. C'est en quelque sorte l'état de l'art des algorithmes biomimétiques du domaine des fourmis artificielles. Nous présentons les techniques traditionnelles de construction d'un graphe de voisinage et consacrons un état de l'art aux techniques de dessin de graphes tout en mettant le focus sur celles qui présentent des capacités interactives et de représentation de la proximité entre les clusters de données.

• Nous proposons la première contribution dans un troisième chapitre en décrivant les études suivantes sur des bases de données numériques, catégorielles (binaires) : une étude expérimentale des propriétés et performances de la méthode initiale CL-Ant. Nous mesurons les indices d'évaluation afin de la qualité de clustering de notre méthode. Nous présentons également l'aspect visuel de graphes à partir de l'algorithme proposé.

• Nous considérons dans un quatrième chapitre une extension nommé CL-AntInc pour la construction incrémentale d'un graphe de voisinage pour lequel nous visualisons les relations de voisinages avec rendu de la proximité réelles entre les clusters de données. Nous avons évalué et validé CL-AntInc sur plusieurs bases de données numériques et binaires. Nous proposons à cet effet une étude expérimentale sur les performances de notre méthode pour des flux de données. Nous présentons l'aspect visuel des graphes construits avec cette méthode.

• Nous consacrons un dernier chapitre à une étude comparative : nous avons comparé l'algorithme initial à une méthode de partitionnement (K-Means),

à une méthode hiérarchique CAH (Classification Ascendante Hiérarchique). Nous comparons également la version incrémentale à des méthodes incrémentales comme CluStream, DenStream et D-Stream en terme d'indices d'évaluation de clustering. Nous présentons les résultats obtenus sur les jeux de données réelles. La complexité de chaque méthode sera présenté également dans ce chapitre.

• A la fin de ce document, une conclusion fait le bilan de l'ensemble de ces travaux et indique les perspectives de développement des méthodes proposées.

Notre travail a fait l'objet des publications scientifiques référencées dans l'annexe A.

CHAPITRE 1

FOURMIS RÉELLES COMME SOURCE D'INSPIRATION

Sommaire

1.1 Introduction 14
1.2 La bio-inspiration et l'intelligence en essaim 15
1.3 Intelligence en essaim, auto-organisation et systèmes complexes 17
1.3.1 L'auto-organisation : concept fondateur de l'intelligence en essaim 17
1.3.2 Les systèmes complexes pour l'intelligence en essaim . . . 20
1.4 Champs de l'intelligence en essaim 22
1.4.1 L'intelligence collective des lymphocytes : système immunitaire 22
1.4.2 Robotique en essaim 22
1.4.3 L'intelligence artificielle distribuée et systèmes multi-agents 25
1.5 Propriétés biologiques des fourmis réelles et application en informatique 27
1.5.1 La division du travail 28
1.5.2 La communication chez les fourmis 28
1.5.3 La construction du nid chez les fourmis 30
1.5.4 Le rassemblement d'objets 31
1.5.5 Auto-assemblage chez les fourmis réelles 32
1.5.6 La stratégie de prédation 33
1.5.7 Reconnaissance chimique des fourmis réelles 34
1.6 Conclusion 36

Résumé

Ce chapitre propose un état de l'art sur les thématiques auxquelles nous nous sommes intéressées pour nos travaux de thèse. Nous avons survolé la notion de l'intelligence en essaim et les principales caractéristiques ainsi que les propriétés que les fourmis réelles possèdent pour la réalisation de tâches complexes. Après une description des sources d'inspirations biologiques, nous proposons un état des lieux de ce qui peut exister en matière de modélisation mathématique inspirée de la biologie pour un certain nombre de développements en informatique.

1.1 Introduction

Au cours de la dernière décennie, l'intelligence en essaim est en fait abordée sous plusieurs angles scientifiques : tout d'abord, en biologie puisque ce domaine reste fortement bio-inspiré, ensuite en informatique où il constitue une branche de l'intelligence artificielle, etc. Les problèmes traités en informatique sont la plupart du temps résolus par des méthodes mathématiques simples. Cependant ces méthodes ne sont pas toujours adaptées pour des problèmes complexes, qui apparaissent par exemple dans le domaine de la robotique, ou de la recherche opérationnelle. Il faut alors inventer de nouvelles méthodes de résolution, et une source d'inspiration utilisée dans la branche de l'intelligence artificielle est la modélisation de systèmes complexes naturels. L'intelligence en essaim s'apparente à une forme collective d'intelligence que l'on observe chez un grand nombre d'espèces animales (insectes, poissons, oiseaux, mammifères, ...). L'avantage de ces approches par rapport aux techniques traditionnelles est leur robustesse et flexibilité. Ces propriétés présentent le succès de l'intelligence en essaim qui est un paradigme pour la conception des algorithmes face à des problèmes complexes en analysant individuellement les capacités cognitives des entités, parfois extrêmement limitées.

Dans cette étude de l'existant, nous allons nous intéresser aux modèles qui copient les comportements sociaux des animaux, c'est-à-dire l'intelligence collective ou l'intelligence en essaim qu'il est possible d'observer plus particulièrement chez les fourmis, qui sera la principale espèce étudiée dans cet état de l'art. D'un point de vue théorique, ce chapitre se situe dans le cadre général de l'étude des systèmes d'intelligence en essaim et ses caractéristiques dans le domaine informatique.

1.2 La bio-inspiration et l'intelligence en essaim

La notion d'intelligence collective est née dans les années 60, de l'application, à la robotique, des analyses des éthologues sur le comportement des sociétés d'insectes. Aujourd'hui, les modèles de comportements collectifs des insectes sociaux fournissent aux informaticiens et roboticiens des méthodes puissantes pour la conception d'algorithme d'optimisation et de contrôle distribué. L'intelligence en essaim a pour objet de transformer les capacités collectives de résolution de problèmes des insectes sociaux en techniques artificielles de résolution de problèmes scientifiques. En plus la capacité des méthodes basée sur l'intelligence en essaim offrent souvent un certain degré de flexibilité et de robustesse dans des environnements dynamiques [Bonabeau 1999a], [Theraulaz 1997] et [Camazine 2001b].

Selon Bonabeau [Bonabeau 1999a], l'intelligence en essaim est née de la modélisation mathématique et informatique des phénomènes biologiques rencontrés en éthologie sur la base d'études statistiques comportementales chez les espèces étudiées. Ces domaines d'applications sont maintenant nombreux pour l'informatique, en recherche opérationnelle notamment et dans les réseaux de communications informatiques.

Bonabeau et al. [Bonabeau 1999a] étend la définition de l'intelligence en essaim au-delà du contexte robotique d'origine :

«L'intelligence en essaim inclut toute tentative de conception d'un algorithme ou d'un dispositif visant à résoudre des problèmes de façon distribuée, inspirée du comportement collectif des insectes sociaux ou d'autres sociétés animales».

Merkle et al. dans l'ouvrage [Blum 2008] propose une définition pragmatique de l'intelligence en essaim du point de vue informatique :

«L'intelligence en essaim est une discipline de l'intelligence artificielle moderne qui traite de la conception de systèmes multi-agents en vue d'applications telles que l'optimisation et la robotique».

«L'intelligence collective désigne les capacités cognitives d'une communauté résultant des interactions multiples entre des membres (ou agents). Les éléments portés à la connaissance des membres de la communauté font qu'ils ne possèdent qu'une perception partielle de l'environnement et n'ont pas conscience de la totalité des éléments qui influencent le groupe. Des agents au comportement très simple peuvent ainsi accomplir des tâches apparemment très complexes grâce à un mécanisme fondamental appelé stigmergie».

Les insectes sociaux sont néanmoins maîtres dans l'art de la construction de

structures gigantesques et de la résolution des problèmes à l'échelle de leur société toute entière [Camazine 2001b], [Couzin 2003b], [Sumpter 2006], [Gueron 1993]. Leurs colonies présentent une gamme très variée de comportements collectifs, qui allient l'efficacité à la flexibilité et à la robustesse. Les insectes sociaux accomplissaient des tâches complexes. En effet, bien que ne pouvant être individuellement qualifiés d'intelligents, les membres de ces sociétés sont collectivement capables de réaliser des constructions sophistiquées, de s'adapter à des environnements changeants, de trouver le plus court chemin à une source de nourriture. Ces animaux à l'apparence simple sont ainsi capables de construire des réseaux de transport et d'échange [Theraulaz 2003], [Buhl 2005], [Buhl 2004] à l'intérieur desquels le trafic des individus et du matériel est régulé [Couzin 2003a], [Dussutour 2004], [Burd 2006],[Vittori 2006].
Ils sont également à même de construire des édifices immenses comparés à leur petite taille [Tschinkel 2004], ils disposent de systèmes de division du travail permettant d'allouer de manière dynamique les différentes tâches à effectuer pour le bon fonctionnement de la colonie [Deneubourg 1987, Gordon 1996, Bonabeau 1998, Beshers 2001].

L'étude du comportement animal présente des aspects passionnants. Les foules de piétons forment également d'étonnantes structures créant ainsi des files alternées d'individus se déplaçant dans le même sens [Helbing 2001]. Parmi eux se trouvent les extraordinaires phénomènes collectifs (voir figure 1.1) que l'on observe au sein de nombreuses sociétés animales [Camazine 2001b, Couzin 2003b, Sumpter 2006, Gueron 1993]. La figure 1.1 montre des exemples de comportements collectifs dans les sociétés animales et humaines. (a) Culture sur agar de la bactérie Bacillus subtilis. Le pattern formé par la colonie dépend à la fois du morphotype de la bactérie et des conditions environnementales [Ben-Jacob 2000]. (b) Colonie de chasse chez la fourmi légionnaire Eciton burchelli. (c) Un banc de poissons formé par l'alignement des individus sur une direction commune de déplacement. (d) Un phénomène similaire est à l'origine de la formation des vols d'oiseaux. (e) Migration d'un troupeau géant de gnous. (f) Déplacement d'une foule de piétons dans une rue commerçante. Les individus se séparent en deux flux de directions opposées (repris depuis [Helbing 2001]).

Une intelligence collective suppose, pour apparaitre, que soient réalisées certaines conditions [Bonabeau 1999a] :

1. Chaque individu est autonome : aucun individu ne dirige tous les autres et il n'y a pas de stratégie au niveau global.
2. Le système est constitué d'individus indépendants et assez simples dont les

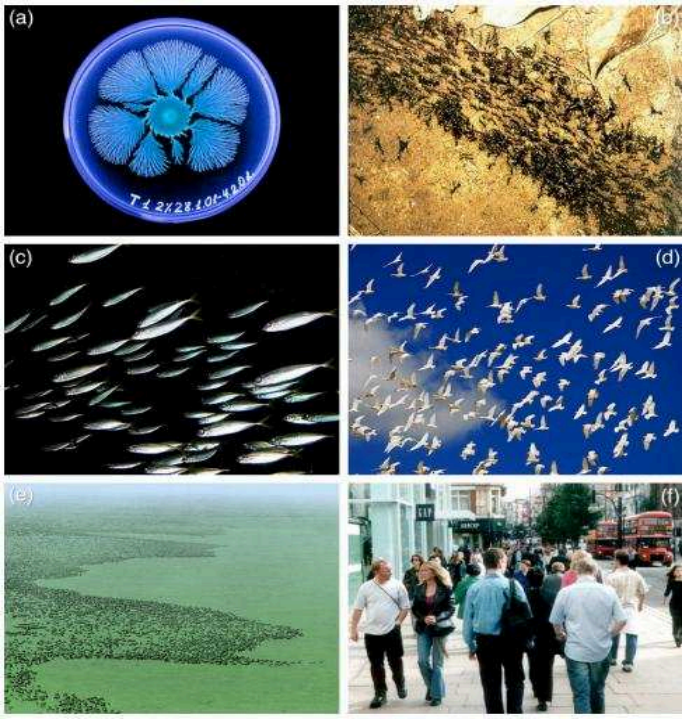


FIGURE 1.1 – Exemples de comportements collectifs dans les sociétés animales et humaines [Garnier 2008].

comportements sont corrélés. Le comportement global est alors le résultat finalisé des comportements individuels.

3. Le système est capable de s'adapter et de résoudre des tâches génériques d'une manière intelligente. Ce comportement émerge des interactions entre individus.

1.3 Intelligence en essaim, auto-organisation et systèmes complexes

1.3.1 L'auto-organisation : concept fondateur de l'intelligence en essaim

Le processus d'auto-organisation est à l'origine de ce que l'on dénomme l'intelligence collective. L'intelligence collective possède des capacités cognitives collectives évoluées qui émergent de capacités cognitives individuelles limitées. Le concept d'auto-organisation est essentiel pour comprendre ces phénomènes Hermann Haken [Haken 2008] définit l'auto-organisation par la manière suivante :

«L'auto-organisation est la formation spontanée, souvent avec un objectif apparent, de structures spatio-temporelles, temporelles, spatiales ou de fonctions dans les systèmes constitués de peu ou de nombreux composants. En physique et biologie, l'auto-organisation apparait dans les systèmes ouverts loin de l'équilibre thermodynamique».

Dans [Théraulaz 1997], les auteurs donnent la définition suivante de l'auto-organisation :

«Auto-organisation est tout processus au cours duquel des structures émergent au niveau collectif (ou plus généralement l'apparition d'une structure à l'échelle N + 1 à partir d'une dynamique définie à l'échelle N), à partir de la multitude des interactions entre individus, sans être codées explicitement au niveau individuel».

1.3.1.1 Mécanismes de l'auto-organisation biologique

L'intelligence en essaim est un domaine fortement bio-inspiré, ce processus de l'inspiration biologique passe naturellement par la connaissance de l'analyse biologique des mécanismes impliqués dans un phénomène naturel. En biologie animale, les phénomènes d'auto-organisation sont les plus spectaculaires et facilement observables (colonies de fourmis, nuées d'oiseaux, bancs de poissons, etc). Les biologistes s'accordent sur la définition de l'auto-organisation [Camazine 2001b] suivante :

«L'auto-organisation biologique possède des mécanismes qui le justifient : parmi les mécanismes responsables de l'auto-organisation biologique, on retrouve des mécanismes fondamentaux mis en exergue en cybernétique, à savoir les mécanismes de rétroactions positives et négatives considérées comme des modes d'interaction entre entités formant le système biologique [Camazine 2001a].»

La possibilité d'échange d'informations, par le biais de modifications de l'environnement, se nomme la stigmergie [Grassé 1959, Theraulaz 1995]. Elle est un concept né des études de Grassé en 1959 sur la construction de termitières [Grassé 1959]. On peut différencier en fait deux classes principales de stigmergie [Bonabeau 1999a]. La stigmergie qualitative, par exemple : la présence d'un trou dans une surface plane sera comblée pour retrouver la planéité de la surface, la planéité (nouveau stimulus) déclenche à son tour la construction de cellules pour l'élevage des larves de la colonie. La stigmergie à base de phéromone est considérée comme quantitative : elle ne change a priori pas de nature au cours de la stimulation et le comportement de la colonie, est gouverné par la concentration de phéromone présente sur l'objet de construction. Ces mécanismes biologiques fondamentaux étant décrits, les principales propriétés des phénomènes biologiques d'auto-organisation sont les suivants [Camazine 2003] :

- La dynamique des processus, dont les éléments constitutifs sont en interaction permanente pour maintenir l'auto-organisation du système. Ces processus dynamiques sont de nature non linéaire ;
- Des propriétés nouvelles émergentes, non présentes au sein des parties élémentaires et non simplement liées à leurs propriétés élémentaires (formation d'amas chez certaines larves d'insectes par exemple). De plus ces processus peuvent mettre en oeuvre des mécanismes localement simples mais qui produisent à grande échelle des résultats complexes ;
- Présence des paramètres : certains paramètres liés au fonctionnement biologique (la composition chimique d'une phéromone par exemple) et d'autres de nature physique (par exemple la volatilité d'une phéromone, la température, son évaporation) peuvent avoir un fort impact sur les processus d'auto-organisation. Cette dépendance paramétrique renforce aussi le rôle et l'importance de l'environnement.

L'auto-organisation se prête bien à l'étude des insectes sociaux qui montrent des comportements collectifs complexes issus de comportements individuels simples. On peut regrouper les processus d'auto-organisation chez les insectes sociaux en quatre groupes tant leur diversité est importante :

- Le recrutement et l'exploitation collective de sources de nourriture : le fourragement met à jour des stratégies qui permettent aux insectes une grande adaptation à leur milieu ;
- La division du travail et l'organisation des rôles sociaux à l'intérieur d'une même société, on peut observer différentes castes spécialisées dans un certain nombre de taches (élevage du couvain, recherche de nourriture, construction du nid, etc) ;
- L'organisation de l'environnement : la construction du nid est un symbole de l'organisation distribuée des insectes. Le nid est construit sans que les insectes soient dirigés, ils répondent à un certain nombre de stimuli provenant de leur environnement ;
- La reconnaissance inter-individuelle : chaque fourmi est capable d'identifier ses congénères tout en participant elle même à l'identité de sa colonie (l'échange d'aliments entre les individus d'une même colonie, la trophalaxie, est un exemple d'acte altruiste permettant en plus d'homogénéiser l'identité de la colonie, l'identité coloniale). Les explications du mécanisme de reconnaissance ne sont pas encore parfaitement établies. Cependant, il s'avère qu'il y ait à la fois une composante génétique et une composante acquise par apprentissage. Un réseau de neurones a par exemple été utilisé pour reproduire ce mécanisme d'apprentissage puis de différenciation entre les compo-

sés chimiques, dans le cas des termites [Bagnères 1998].

1.3.1.2 Auto-organisation en intelligence artificielle

Les premiers champs de l'informatique à avoir exploité les phénomènes d'auto-organisation sont les algorithmes évolutionnaires [Rechenberg 1973], [Beyer 2002], [Holland 1992]. L'intérêt de l'approche informatique tient à son utilisation extensive de la notion d'agent qu'elle développe par ailleurs et qui est particulièrement adaptée à l'étude des phénomènes d'auto-organisation. L'auto-organisation est un phénomène décrit dans plusieurs disciplines, notamment en physique [Nicolis 1977], [Prigogine 1971] et en biologie [Camazine 2001a].

Une définition a été proposée par [Camazine 2001a] :
«L'auto-organisation caractérise un processus au cours duquel une structure émerge au niveau global uniquement d'un grand nombre d'interactions entre les composants de niveau local du système. De plus, les règles spécifiant les interactions entre composants du système sont suivies en utilisant uniquement des informations locales, sans référence a la structure globale.»
Stephanie Forrest utilise la notion d'agent pour définir la computation émergente [Forrest 1991] qui est constituée par les éléments suivants :

- Un ensemble d'agents programmés explicitement : un micro-programme local à l'agent définit son comportement localement (typiquement les automates cellulaires) ;
- Des interactions entre ces agents, réglées par leur propre programme, qui ont pour résultante la formation d'un schéma global (pattern) au niveau macroscopique ;
- L'interprétation naturelle de ces schémas comme du calcul.

Dans cette définition entrent plusieurs domaines de développement de l'informatique : les modèles connexionnistes, les automates cellulaires, les modèles biologiques, les systèmes classifieurs, les modèles de vie artificielles, et les modèles de coopération dans les systèmes sociaux sans autorité centralisée. Forrest [Forrest 1991] insiste sur le fait que la computation émergente concerne des processus de calcul non linéaires. La non linéarité est communément considérée comme caractéristique dans le formalisme mathématique de description des systèmes complexes

1.3.2 Les systèmes complexes pour l'intelligence en essaim

L'intelligence en essaim constitue désormais un domaine à part entière de l'intelligence artificielle distribuée. Les problématiques qu'elle soulève touchent ce-

pendant à de nombreux autres domaines scientifiques. En particulier le concept d'essaim trouve pleinement sa place au sein de la science dites des «systèmes complexes».

Plusieurs auteurs ont défini les systèmes complexes de façon variable, mais on retrouve systématiquement une définition commune : un système complexe est constitué d'un grand nombre d'entités en interaction mutuelle, et éventuellement en interaction avec un environnement, dont le comportement global émerge de façon non linéaire du niveau local. Pour certains auteurs, la non linéarité est essentielle pour exprimer la complexité même si le nombre d'entités est faible.

L'intelligence en essaim vise à fournir une grandeur quantifiable de l'auto-organisation des systèmes. Une science s'est développée autour de cette problématique portant le nom de science des systèmes complexes qui a pour objectif de théoriser les phénomènes d'auto-organisation. La question cruciale est donc de comprendre comment les composants d'un système émergent entre eux pour produire une structure complexe. Rodolphe Charrier [Rodolphe 2009] présente ainsi la conception, les caractéristiques et les applications d'un modèle original, le système multi-agent logistique (SMAL), pour le domaine de l'intelligence en essaim. Le SMAL trouve son origine en modélisation des systèmes complexes : il est en effet issu des réseaux d'itérations logistiques couplées qui est adapté dans le modèle de calcul au schéma «influence-réaction» des systèmes multi-agents.

Nino Boccara dans la préface de son ouvrage dédié aux systèmes complexes [Boccara 2004] propose la définition à base d'agents suivante :

«Bien qu'il n'existe pas de définition universellement acceptée pour définir un système complexe, la plupart des chercheurs décrirait comme complexe un système d'agents liés les uns aux autres qui exhibent un comportement global émergent non pas imposé par un contrôleur central, mais résultant des interactions entre les agents. Ces agents peuvent être des insectes, des oiseaux, des personnes, ou des entreprises, et leur nombre peut varier d'une centaine à un million».

Les systèmes complexes présentent des types de comportement qui sont dans une large mesure indépendante de la nature de l'élément individuel, on peut citer par exemple :

- Les étoiles dans une galaxie,
- Les composantes d'une atmosphère planétaire,
- Les habitants d'une ville,
- Les voitures dans un embouteillage de la circulation,
- Les insectes sociaux dans une colonie,
- Les courants dans un liquide en mouvement,
- Les cellules dans un organisme multicellulaire,

– Les gènes dans un réseau,
– Les groupes de molécules dans un milieu réactif,
– Les atomes magnétiques dans un verre de spin.

1.4 Champs de l'intelligence en essaim

1.4.1 L'intelligence collective des lymphocytes : système immunitaire

Le système immunitaire est responsable de la protection de l'organisme contre les «agressions» d'organismes extérieurs. La réponse immunitaire est basée sur le principe de la sélection clonale, appelé aussi, théorie du choix clonal [De Castro 2000a]. L'immunologie a pour objet, l'étude du système immunitaire. Ce système, composé d'organes, de cellules et de molécules, assure le maintien de l'intégrité de l'organisme qu'il défend. Ainsi, il peut être défini comme étant, l'ensemble des mécanismes biologiques permettant à l'organisme de reconnaître et de tolérer ce qui lui appartient en propre (le soi), et de reconnaître et de rejeter ce qui lui est étranger (le non soi) : les substances étrangères ou les agents infectieux auxquels il est exposé, mais aussi, ses propres constituants altérés (comme les cellules tumorales). Ce mécanisme de reconnaissance intelligent a été modélisé pour donner naissance au système immunitaire artificiel AIS (Artificial Immune System) [Timmis 2000] et [Timmis 2001].

Ces systèmes disposent de propriétés complexes car ils sont capables de générer des solutions et de les sélectionner en fonction de leur efficacité selon des heuristiques originales [Azzag 2000]. Une description des fondements théoriques et de nombreuses applications des systèmes immunitaires artificiels peut être trouvée dans [De Castro 1999], [Dasgupta 1997] et [Dasgupta 1999]. Le principe des systèmes immunitaires est appliqué au problème de classification [De Castro 2000b]. D'autres modèles plus complexes existent. Ainsi dans [Timmis 2002] le système utilise plusieurs niveaux de cellules et d'interaction (anticorps, lymphocytes, cellules de mémorisation). Dans [Nasaroui 2002], ce même système est généralisé et amélioré en utilisant des fonctions d'appartenance floue plutôt qu'une distance euclidienne et un seuil.

1.4.2 Robotique en essaim

1.4.2.1 De la robotique classique vers la robotique collective

La robotique classique, comme l'intelligence artificielle classique, a conduit au développement de processus et d'entitées sophistiquées aux capacités voisines

de celles de l'homme. Plus récemment, comme pour l'intelligence artificielle distribuée, l'intérêt de développer des collectifs de robots interagissant s'est développée. Un des avantages de ces systèmes décentralisés réside dans le fait que le contrôle est décentralisé en tâches simples sur chaque robot coopérant plutôt que dans un super-robot pour lequel le modèle même du contrôle est parfois difficile à concevoir. Un contrôle distribué, à l'inverse, associe les activités au niveau local, chaque robot n'ayant qu'un dynamisme et une idée limités : c'est la robotique collective. La robotique en essaim est un champ de recherche relativement nouveau qui s'intéresse à la coordination auto-organisée de systèmes multi-robots, et tout particulièrement à ceux qui consistent en un grand nombre d'unités dont la complexité est minimisée. Originellement inspiré par des exemples naturels, principalement dus aux insectes sociaux [Bonabeau 1999a], le champ de la robotique en essaim applique les principes de l'intelligence collective aux systèmes robotiques, dans le but de tirer parti de comportement globaux complexes émergeant des interactions locales entre les agents et l'environnement, perturbées par le bruit et la portée limitée de leurs moyens de perception, de communication et d'actuation. La robotique en essaim cherche en effet à se démarquer de la robotique distribuée ou collective par sa bio-inspiration explicite. Une définition du domaine est donnée dans [Dorigo 2004], [Sahin 2008] :

«La robotique en essaim est l'étude de la façon dont un grand nombre d'agents relativement simples incarnés physiquement peuvent être conçus pour qu'un comportement voulu puisse émerger des interactions locales entre les agents et entre les agents et l'environnement».

La robotique en essaim considère des robots en général identiques, de conception simple qui ont des perceptions et interactions uniquement locales. Ces essaims de robots sont susceptibles d'offrir les caractéristiques suivantes qui rejoignent celles de la robotique collective [Sahin 2008] :

- **La robustesse :** la redondance inhérente au système, la coordination décentralisée, la simplicité de conception de chaque entité et la distribution de la perception, font des essaims de robots des systèmes robustes aux pannes et aux variations de l'environnement ;
- **La flexibilité :** concerne la capacité des essaims de robots à traiter différents types de problèmes ;
- **Scalabilité :** les performances ne sont globalement pas impactées par la taille de l'essaim.

Actuellement des recherches sont menées dans le sens du développement de systèmes intelligents de robots inspirés de l'étude des sociétés d'insectes. Les ré-

seaux neuronaux constituent un exemple d'organisation de petites unités réactives simples (les neurones artificiels) de l'interaction desquelles peuvent émergé un comportement global «intelligent». L'équipe du biologiste Danois Axel Michelsen a conçu Robobee en 1989 [Michelsen 1989] comme montre la figure 1.2 une bille de bronze et de cire de la taille d'un insecte, animée par une tige métallique et dotée, en guise d'aile, d'une larme de rasoir actionnée par un électroaimant. Capable de produire la danse des abeilles butineuses indiquant la localisation de zones nourricières, Robobee a réussi à diriger les abeilles d'une ruche vers des objectifs prédéterminés. Elaborer des «leurres intelligents», capables de récupérer de l'information dans la collectivité pour interagir avec elle et mieux la manipuler est justement l'objectif du projet européen Leurre.

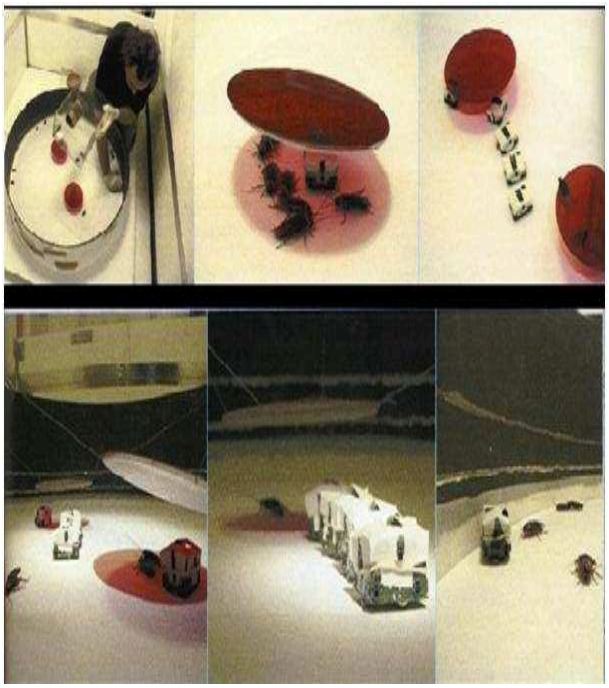


FIGURE 1.2 – Le robot nommé «Robobee» : leurres intelligents [Michelsen 1989].

Guy Théraulaz est le précurseur en France d'une nouvelle approche : «La robotique collective constitue une alternative à l'utilisation systématique d'un robot unique, ultrasophistiqué, muni de nombreux capteurs et d'algorithmes très complexes de navigation et de prise de décision».
Craig W. Reynolds [Reynolds. Flocks 1987] a été le premier à concevoir de tels systèmes dans le domaine de la Synthèse d'Images. La gestion de populations d'individus (bancs de poissons, vols d'oiseaux, colonies d'insectes, ...) n'est plus confiée à un algorithme déterministe et séquentiel, mais est le résultat des nombreuses interactions entre les individus, chacun doté de capacités perceptives lo-

cales et limitées, mais dont le comportement global exhibe une complexité et une apparente motivation que l'on ne trouve que dans les sociétés naturelles.

Les types de problèmes à résoudre dans la robotique en essaim sont pour beaucoup issues des phénoménologies identifiées dans l'auto-organisation biologique [Sahin 2008], [Beni 2005], [Dorigo 2005] :

- Agrégation et dispersion sont les deux faces d'une même pièce : l'essaim doit pouvoir explorer son environnement (dispersion) et également se regrouper pour exploiter (agrégation) ;
- Fourragement : inspiré des comportements fourmis, l'essaim doit être capable d'optimiser sa recherche de ressources ;
- Auto-assemblage et mouvement coordonné : l'essaim doit pouvoir se souder pour constituer une structure répondant à une contrainte du terrain (ex : formation de pont chez les fourmis, tractage d'une proie, ...). Dans cette configuration où les robots sont reliés physiquement, l'objectif est de pouvoir continuer à se mouvoir collectivement ;
- Transport coopératif et collectif : la métaphore ici est celle de fourmis transportant une proie à plusieurs individus ;
- Génération de formes et auto-organisation : cela correspond à la capacité de l'essaim à produire des schémas de regroupement spécifiques ou singuliers par processus auto-organisé.

La robotique collective auto-organisée s'appuie sur une logique décentralisée utilisant des unités disjointes, simples et aléatoires, distribuées dans l'environnement sans une connaissance exhaustive de celui-ci. Ces unités interagissent avec l'environnement et entre elles au moyen de signaux dont seule l'intensité est signifiante. Chaque agent est capable d'anticiper une action en extrapolant sa situation actuelle au voisinage dont il a connaissance. Dans de tels systèmes, la solution au problème posé n'est pas programmée, mais émerge des nombreuses interactions. La résolution collective d'une tâche fait appel à la coopération. L'intelligence Artificielle Distribuée s'est orientée vers des systèmes multi-agents dont les éléments intégraient des comportements sophistiqués, la Robotique Collective, s'inspirant de l'intelligence collective, n'utilise que des agents simples (sur le mode stimuli-réponse) n'ayant pas de représentation du système dans son ensemble.

1.4.3 L'intelligence artificielle distribuée et systèmes multi-agents

Le principe de l'intelligence collective de plusieurs entités simples en interaction desquelles émerge à un niveau collectif une structure complexe (qualifiée d'intelligente) est présent dans différentes disciplines : biologie, physique, informatique, etc. Dans ces domaines, différents modèles ont vu le jour pour décrire et analyser de tels systèmes. En informatique, les systèmes multi-agents réactifs

FIGURE 1.3 – (a) Le robot Genghis développé par Rodney Brooks [Brooks 1989]. (b) Le robot Sahabot développé par Lambrinos et al. [Lambrinos 2000] pour tester les hypothèses sur la navigation de la fourmi Cataglyphis. (c) William Gray Walter et son robot ELSIE rentrant à son poste de rechargement [Holland 1992]. (d) Le robot Insbot développé dans le cadre du projet européen Leurre pour interagir avec un groupe de blatte Periplaneta americana [Halloy 2007].

offrent un cadre conceptuel permettant la représentation et la simulation de tels systèmes. Une manière de concevoir de tels systèmes est donc d'effectuer un lien entre comportement individuel et réponse collective est de s'inspirer des phénomènes existants en biologie, notamment dans le cadre des sociétés d'insectes. Une telle approche fournit des solutions dont les caractéristiques essentielles sont :

- **Dynamique :** le système s'organise en fonction de son contexte courant et il est capable de s'adapter continuellement aux variations de celui-ci ;
- **Décentralisé :** il n'y a pas besoin d'une entité centralisatrice dictant ce que doit faire chaque agent, on s'affranchit des problèmes de goulot d'étranglement, de construction et maintenance d'une représentation globale ainsi que des problèmes de dysfonctionnement de cette entité centralisatrice ;
- **Simple :** le modèle individuel est relativement simple et nécessite des efforts de conception moindre.

Dans le cadre de l'organisation de système multi-agents, envisager un collectif pose alors le problème d'organiser les différentes activités des agents afin que globalement ce collectif se comporte comme un tout cohérent et réponde aux exigences qui lui sont fixées en fonction des capacités propres à chacun, des caractéristiques changeantes de leur environnement, etc. Ce problème d'organisation fait l'objet de nombreuses études dans le cadre des systèmes multi-agents.

- Pour **les agents cognitifs** ceux-ci sont dotés de moyens de représentation, de raisonnement et de communication élaborés. Ils peuvent ainsi représenter la tâche à résoudre, les compétences des autres, raisonner sur les conséquences de leurs actions et interagir de manière élaborée avec les autres ;
- Dans **le cas réactif** les agents n'ont pas ou peu de représentation des autres, ils possèdent des mécanismes de décision de type stimulus-réponse et interagissent via leur environnement avec les autres. Ils ne disposent que d'informations locales. Dans ce second cas, les agents ne peuvent être réellement qualifiés d'intelligents individuellement mais la société qu'ils forment peut l'être. On parle alors de systèmes multi-agents réactifs ou d'intelligence en essaim.

1.5 Propriétés biologiques des fourmis réelles et application en informatique

Les différentes problématiques posées par les systèmes biologiques sont une source importante d'inspiration pour la conception des systèmes artificiels. Millonas [Millonas 1993] a tiré de la métaphore fourmi une proposition de caractérisation en cinq principes des systèmes d'intelligence en essaim.

- **Principe de proximité** : ce premier principe pose la nécessité d'une capacité des entités du groupe à répondre dans une localité spatiale et temporelle aux signaux/stimuli de l'environnement. Cela implique une capacité de calcul local, calcul dont l'objectif est de décider d'un comportement qui maximiserait au mieux l'utilité pour l'activité du groupe entier (par exemple le fourragement, la construction d'un nid, le déplacement de groupe,...),
- **Principe de qualité** : le groupe doit pouvoir répondre aussi à des critères de qualité par rapport à l'objectif recherché (qualité de la source de nourriture, de l'emplacement du nid, ...). Il s'agit là d'optimiser un critère de qualité pour l'essaim,
- **Principe de réponse diversifiée** : le groupe doit pouvoir répondre dynamiquement aux éventuels changements de l'environnement, et donc diversifier

autant que possible ses modes de fonctionnement,
- **Principe de stabilité** : ce principe vient compléter et limiter le précédent. Le groupe doit également maintenir une forme de stabilité afin d'éviter de basculer d'un mode de fonctionnement à l'autre au moindre changement de l'environnement, ce qui serait une perte d'énergie considérable,
- **Principe d'adaptabilité** : le changement de mode de fonctionnement, lorsque le mode courant n'est plus suffisamment satisfaisant, s'opère par adaptation à une nouvelle situation environnementale.

1.5.1 La division du travail

Une caractéristique fondamentale des insectes sociaux est la division du travail. Dans le cadre de la division du travail, la fréquence des interactions entre les individus dépend de leur âge, de la fonction qu'ils remplissent dans la colonie, et des déplacements associés. Les tâches que doivent accomplir les ouvrières sont en effet multiples :

- La récolte du nourriture,
- L'entretien et défense du nid,
- La construction du nid,
- L'élevage des couvains,
- L'entretien des larves et leur approvisionnement en nourriture.

Les chercheurs ont essayé de relier les comportements des ouvrières à leurs caractéristiques individuelles. Le premier paramètre caractérisant les ouvrières est leur morphologie. La variation de taille peut être continue ou discontinue, donnant alors des castes distinctes. L'âge des ouvrières est un autre facteur important dans la division du travail. Les jeunes ouvrières assurent le soin au couvain au centre du nid, puis remplissent peu à peu des tâches plus périphériques, comme l'entretien du nid et occupent à la fin de leur vie le rôle de fourrageuses. Ces deux facteurs, âge et morphologie, n'expliquent cependant qu'une partie de la division du travail entre les ouvrières. Il a été mis en évidence que certains groupes d'individus se spécialisent dynamiquement pour une tâche particulière [Grassé 1939]. Toutes ces activités, dont l'importance est variable dans le temps et l'espace, doivent être assurés simultanément pour la survie et le développement de la colonie. C'est essentiellement la plasticité de l'organisation déployée par les fourmis qui nous intéresse.

1.5.2 La communication chez les fourmis

Les insectes sociaux en général, et les fourmis en particulier, ont développé des mécanismes de communication très élaborés (voir [Breed 1998] pour les insectes sociaux et pour les animaux en général [Brossut 1996]). On distingue dans la com-

munication quatre types de signaux selon leur nature : sonore, tactile et chimique. Il a été défini quelques types de réponse mettant en oeuvre une forme de communication [Holldobler 1990] : l'alarme, le recrutement (pour une source de nourriture ou un site de nidification), la reconnaissance des apparentés ou de caste, le marquage du territoire et du nid, et la reproduction (diffiérenciation du sexe, de l'espèce, de la colonie...). On connaît quand deux fourmis se rencontrent, elles communiquent entre elles avec leurs antennes. En fait elles se touchent pour échanger des informations sur leur visa chimique. Chez les insectes, la communication chimique est fondamentale. Les phéromones (mélange d'hydrocarbures) sont à la base de la communication de nombreuses espèces. Les différentes applications informatiques qui découlent des capacités de communication des fourmis se retrouvent par exemple en optimisation combinatoire où la coopération stigmergétique s'applique parfaitement à la recherche du plus court chemin dans un graphe.

Le déplacement grâce à des repères chimiques est une des originalités comportementales que l'on retrouve des mammifères (ex : rat [Galef 1996]) aux unicellulaires (ex : [Burchard 1982]) en passant par les insectes sociaux (fourmis : [Holldobler 1990], termites : [Grassé 1959]). Ces substances chimiques communément appelées phéromones sont libérées par un animal dans son milieu, et vont affecter le comportement ou la physiologie des individus de la même espèce [Karlson 1958, Passera 1958]). Il existe différents types de phéromones. Les phéromones qui conduisent à un déplacement collectif d'individus peuvent être soit des phéromones d'alarmes soit des phéromones de pistes. De nombreux algorithmes sont inspirés du comportement de masse des fourmis pour la recherche de nourriture : les phéromones de pistes. 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 finira pas être emprunté par presque 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. Les fourmis peuvent également émettre des phéromones d'alarmes, grâce à des glandes situées près des mandibules. Ce sont des molécules très volatiles qui induisent un comportement agressif chez leurs congénères : les ouvrières prennent alors des attitudes menaçantes, et attaquent l'ennemi. 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 (du à l'évaporation), elles attaquent toutes ensemble la proie.

1.5.3 La construction du nid chez les fourmis

L'une des manifestations les plus fascinantes de l'auto-organisation des sociétés d'insectes est leur capacité à construire collectivement des nids dont l'architecture peut également être très complexe. Les insectes organisés en sociétés comme les abeilles, fourmis, guêpes et termites sont capables de construire collectivement des nids d'une remarquable complexité. Ces productions collectives cohérentes ne pourraient exister sans une forme de coordination entre les individus. Guy Theraulaz et al. [Theraulaz 1995], prenant pour exemple la construction collective chez les guêpes, ont développé un modèle numérique permettant de reproduire les différentes phases de construction d'un nid. Dans ce modèle, les insectes sont représentés par des agents qui obéissent à des règles simples et n'interagissent pas directement entre eux. Il apparaît dans le cadre de ce modèle, que seuls certains enchaînements de comportements possédant des propriétés spécifiques produisent des architectures cohérentes et que l'ensemble de ces programmes correspond à un répertoire fort restreint de formes. Les architectures obtenues, correspondant à celles produites par les sociétés de guêpes, tendent à démontrer que pour fabriquer un nid collectif, ces insectes n'ont pas besoin de communiquer directement entre eux : ils se coordonnent à travers l'architecture qu'ils construisent.

Dès la fin des années 1950, Pierre-Paul Grassé [Grassé 1939] avait montré que, chez les termites, la régulation de l'activité bâtisseuse ne dépendait pas directement des insectes, mais des constructions elles-mêmes. Il a étudié les différentes étapes de la reconstruction du nid chez les termites et il s'agit là sans doute d'une des premières observations de l'ordre par fluctuations. Ses études l'ont conduit à formuler la théorie de la stigmergie qui explicite l'intersection animal-travail. Deneubourg [Deneubourg 1977] a étudié la construction du nid chez les termites. L'apparition des piliers dans une termitière pouvant être expliquée par l'amplification de multiples fluctuations chaotiques : la structure, modèle d'équilibre des forces par sa stabilité, nait de l'amplification de multiples déséquilibres. La construction des nids de guêpes est aussi un modèle d'action collective où les agents répondent plus particulièrement à des stimuli issus de certaines configurations dans la structure [Bonabeau 1999a]. Ainsi les fourmis construisent collectivement des nids dont la taille importante en comparant avec celle des individus et à l'architecture parfois très complexe. Leur capacité à coordonner plusieurs milliers d'individus pour bâtir leurs nids.

1.5.4 Le rassemblement d'objets

Le tri du couvain est l'un des exemples d'intelligence collective qui représente un phénomène de l'auto-organisation [Camazine 2001a], observé chez certaines colonies de fourmis (lasius niger [Depickére 2004], leptothorax [Franks 1992], Pheidole pallidula ou Messor sancta [Deneubourg 1991]). Ces colonies organisent le couvain en un unique amas de larves, et, notamment chez la fourmi leptothorax unifasciatus, ce couvain est constitué d'anneaux concentriques rassemblant des larves de types différents : les plus petites au centre et les plus grosses en périphérie. Il est important de noter que cette structure est utile à la collectivité, en permettant par exemple aux larves matures d'être nourries en priorité [Franks 1992].

Dans la nature, les fourmis offrent un modèle stimulant pour le problème du partionnement. L'exemple du tri collectif du couvain ou de la constitution de cimetières sont les plus marquants. Certains travaux expérimentaux montrent que certaines espèces de fourmis sont capables d'organiser spatialement divers éléments du couvain : les oeufs, les larves et les nymphes [Deneubourg 1990, Franks 1992]. Dans un article de Bonabeau [Bonabeau 1999a] paru dans la revue de vulgarisation scientifique «Pour la Science», l'algorithme du tri du couvain est décrit ainsi : «Chez les fourmis Messor sancta, les ouvrières nettoient les nids en entassant à l'extérieur les cadavres. Dans des nids artificiels reproduits en laboratoire, elles regroupent en quelques heures des corps dispersés de façon aléatoire [...]. De la même manière, les ouvrières de l'espèce Leptothorax unifasciatus trient systématiquement les larves et les oeufs. Les petites larves sont regroupées avec les oeufs, les pupes et les prépupes (des stades intermédiaires dans le développement des insectes) sont autour, elles-mêmes entourées par les larves les plus grandes». Les fourmis réelles offrent un modèle stimulant pour le problème du partitionnement et de la classification. La constitution de cimetières, le tri collectif du couvain quand il est dérangé, et le rassemblement des oeufs en fonction de leur état de développement, sont les exemples les plus marquants.

Les premiers modèles informatiques sont ceux de [Lumer 1994b]. D'autres travaux ont été proposés [Monmarché 2000b]. L'auteur propose ainsi, avec l'algorithme AntClass, de découvrir automatiquement un ensemble de classes dans des données numériques sans en connaître le nombre a priori. La figure 1.4 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.

Selon Deneubourg, de petits amas de cadavres s'agrandissent, car ils attirent les ouvrières qui y déposent plus d'éléments : cette rétroaction positive entraîne la formation de tas de plus en plus grands. Pour les couvées, les fourmis ramassent et déposent les éléments en fonction du nombre d'objets similaires environnants. Par exemple, quand une fourmi trouve une grande larve entourée d'oeufs, elle prend de

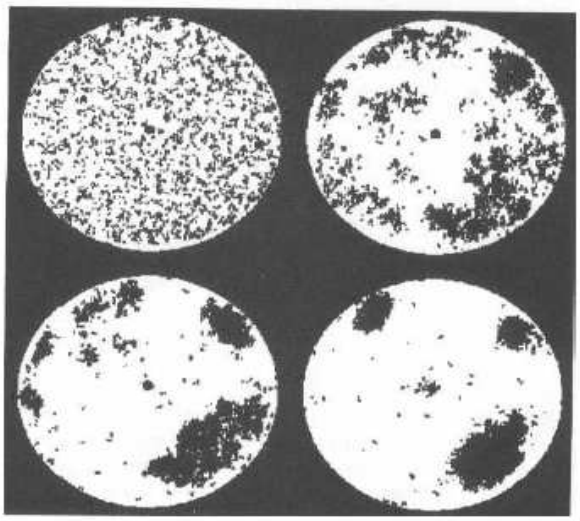


FIGURE 1.4 – Expérience du tri du couvain chez les Messor sancta. 4 images prises (de gauche à droite) à l'état initial, 2 heures, 6 heures et 26 heures après le début de l'expérience [Khedam 2008].

préférence la larve qui constitue «l'intrus» et la dépose dans une zone qui contient déjà d'autres grandes larves.
Ce tri du couvain, en comparant à d'autres constructions réalisées par les insectes sociaux, présente la complexité des relations entre les comportements individuels et collectifs. C'est en grande partie pour cela que ce phénomène a influencé le développement de domaines tels que les algorithmes de colonies de fourmis [Dorigo 2006, Lumer 1994b, Kuntz 1998] ou la robotique collective [Melhuish 2001].

1.5.5 Auto-assemblage chez les fourmis réelles

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 chaine, c'est-à-dire qu'elles s'accrochent entre elles (elle s'auto assemblent) puis courbent et rapprochent les feuilles. La couture est ensuite faite avec la soie collante produite par les larves. 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.

Dans [Anderson 2002], les auteurs ont regroupé plusieurs phénomènes d'auto-assemblage que l'on peut observer chez les animaux, en particulier les insectes

sociaux. La formation de chaînes et de ponts chez les fourmis fileuses Oecophylla permet le franchissement d'obstacles naturels ou sert à la construction de nids. Quant aux fourmis Linepithema humile elles construisent des grappes douvrières suspendues dans le vide dont la fonctionnalité est méconnue à ce jour. Ce phénomène de croissance des chaînes ou la construction de grappes fait appel à des comportements de base tout à fait similaires : des ouvrières gagnent la structure collective (chaîne ou grappe), et y séjournent un certain temps au cours duquel elles se déplacent, éventuellement s'immobilisent avant de quitter cette structure. Ce décrochage se traduit par le phénomène de résorption de chaînes observé chez les Oecophylla longinoda et la chute de gouttes de fourmis observé chez les Linepithema humile. Un état de l'art plus complet est disponible sur le sujet dans [Azzag 2005]. La robotique se trouve être un domaine d'inspiration pour la mise en oeuvre des principes d'auto-assemblage, un survol sur l'auto-assemblage en robotique est déjà disponible dans [Lavergne 2008b]. Les fourmis ont inspiré les scientifiques dans plusieurs domaines, l'optimisation et la classification. Cette notion est plus particulièrement portée sur les modèles de classification.

1.5.6 La stratégie de prédation

Des chercheurs se sont intéressés à la manière dont les fourmis chassent leurs proies et ont proposé de modéliser des stratégies de prédation sans phéromones pour résoudre des problèmes d'optimisation. On peut en trouver un exemple chez certaines fourmis, par exemple les Anomma chasseresses d'Afrique. Elles forment des colonies de plus de 20 millions d'individus et commencent par se répandre en nappe au-dessous de l'arbre qu'elles vont prospecter. Puis une colonne monte à l'assaut. En progressant depuis le tronc jusqu'à l'extrémité des branches les plus minimes, elle va en chasser toute la faune de l'arbre : les proies se laissent généralement tomber sur le sol où elles sont capturées par les fourmis. Les fourmis rousses, en revanche, dont la seule source de protéines est constituée par la viande des insectes, ne paraissent pas posséder de stratégie particulière : il semble qu'elles appréhendent tout ce qui bouge ou même tout ce qui peut se transporter y compris des débris minéraux ; mais des équipes de trieuses rejettent hors du nid ce qui ne convient pas. Un autre type de stratégie se rencontre chez les «fourmis cadavres» comme Paltothyreus, remarquables par l'odeur insupportable qu'elles dégagent. Elles se nourrissent particulièrement de termites. Cet envoi préalable d'éclaireuses se rencontrerait aussi chez les fourmis esclavagistes qui vont récolter dans les nids d'autres espèces les nymphes qui, une fois adultes, leur serviront d'esclaves.

D'un point de vue biologique, ces fourmis vivent dans de petites colonies comportant rarement plus d'une centaine d'individus et établissent leur territoire d'exploration autour de leur nid en le partitionnant en sites de chasse. Chaque fourmi s'attribue des sites particuliers qu'elle va explorer aléatoirement en mémorisant

toutefois préférentiellement ceux sur lesquels elle a pu rencontrer un certain succès. Au fur et à mesure de leur sortie, les ouvrières chargées de récolter de la nourriture s'éloignent de plus en plus du nid couvrant petit à petit une grande partie de leur espace de recherche [Fresneau 1994].

D'autres contraintes pouvant avoir un impact dans un algorithme d'optimisation peuvent être mentionnées. Dans [Monmarché 2000] l'auteur a modélisé le comportement des fourmis Pachycondyla apicalis et a repris ces principes pour résoudre des problèmes d'optimisation numériques (par exemple recherche d'un minimum global) et combinatoire (problème du PVC). Ce modèle a aussi été utilisé par [Picarougne 2004] pour la recherche de document sur internet et [Drogoul 1999] pour la construction de robots.

1.5.7 Reconnaissance chimique des fourmis réelles

La reconnaissance coloniale n'est pas connue que chez les fourmis mais aussi pour un nombre d'espèces sociales comme les abeilles, les guêpes ou bien les termites [Carlin 1987], [Breed 1998], [Bagnères 1998]. Cette reconnaissance repose [Dahabi 1998] sur un message chimique qui reflète l'état de la colonie à un instant donné ainsi que sur son partage entre les différents partenaires pour aboutir à une odeur commune à tous les membres de la colonie. Dans le cas des fourmis, leur odeur est identique à celui de leur colonie et du nid à un instant donné. Cette section a pour objet de présenter les principaux acteurs et comportement à l'origine de l'établissement d'une odeur coloniale. Ces derniers sont inclus dans le littérature biologique sur les socétés fourmis.

Les fourmis vivent au sein de sociétés dont la survie est garantie par un mécanisme de fermeture coloniale comparable au système immunitaire chez l'être humain, qui leur permet de protéger leurs nids en reconnaissant et rejetant les intrus. Cette fermeture se manifeste grâce à un système de reconnaissance basé sur des messages chimiques portés par chaque individu comme étant une carte d'identité odorante, servant à l'identifier comme étant un nombre d'un nid par rapport à un autre. Cette odeur est constituée de manière collective par des échanges chimiques répétés continuellement entre les membres de colonie. Les fourmis vivent donc regroupées au sein de colonies, qui, à l'image de super-organismes, doivent pouvoir distinguer les individus leurs appartenant des autres. 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.

Les descriminateurs chimiques sont des substances portées par chaque fourmi et qui vont servir de base au niveau individuel pour savoir si une fourmi rencontrée

doit être acceptée ou rejetée. Ces descriminateurs ont des origines et de la multitude de comportements observés. Cependant, d'après [Carlin 1987], on peut présenter 4 hypothèses qui ont été envisagées dans la littérature, pour décrire l'origine de ces descriminateurs qui représentent l'origine de l'odeur chimique chez les fourmis :

- Ils peuvent être purement spatiaux. Le visa chimique se résume alors à un marqueur territorial qui réflète les essences chimiques issues des matériaux du nids et de son environnement proche. Donc la fourmi reconnaît une intruse, si celle-ci ne possède pas le marqueur territorial adéquat ;
- Ils peuvent être alléliques : c'est l'expression des gènes dans l'odeur serait reconnue de façon innée par les autres fourmis. Cette hypothèse est cependant peu viable, car des expériences ont montré qu'une fourmi qu'on extrait de sa colonie d'origine pour l'intégrer dans une autre, peut s'y attacher au point qu'elle agressera les membres de son ancienne colonie en ayant oublié sa réelle identité ;
- Ils peuvent correspondre à une identification par appariement de phénotype. Celui-ci implique un apprentissage préalable des traits distinctifs de la colonie afin de déterminer par la suite si un individu rencontré est un intrus ou non, en observant la présence ou l'absence de ces mêmes traits ;
- Ils peuvent correspondre à un apprentissage de ce qui est le soi, ici la colonie, en mémorisant les visas chimiques de toutes les fourmis déja rencontrées dans le nid. Dans ce cas, chaque fourmi se détermine individuellement une «image» de ce que doit être l'odeur moyenne de la colonie et des fourmis y apportenant. Cette hypothèse semble être confirmée d'après [Carlin 1987] par les études les plus récentes.

Dans la thèse de Nicolas Labroche [Labroche 2003], toute une étude est élaborée sur l'établissement de l'odeur chimique au niveau moléculaire, en essayant de comprendre les organes en cause dans sa mise en place, son stockage et sa régulation.

Les comportements collectifs des sociétés d'insectes comme les fourmis ont fortement inspiré les scientifiques. L'algorithme AntClust [Labroche 2003] (voir en détails le principe de cet algorithme dans le chapitre suivant) s'inspire du phénomène de reconnaissance chimique chez les fourmis. Il repose sur l'apprentissage et le partage d'une odeur coloniale commune à toutes les fourmis d'un même nid [Labroche 2003], [Maziz 2004].

Selon ces principes nous avons développé un modèle de règles comportementales pour des fourmis artificielles que nous appliquons dans cette thèse à la classification non hiérarchique de données. Nous allons détailler dans les chapitres 3 et 4 ces différents principes pour développer un algorithme de classification non hiérarchique et non supervisée.

1.6 Conclusion

Dans ce chapitre nous avons présenté un court historique du concept d'intelligence en essaim, ses définitions et ses caractéristiques et des différents champs où il a été appliqué. Les avantages de méthodes fondées sur l'intelligence en essaim sont à la fois leur flexibilité (elles s'adaptent aux modifications de l'environnement) et leur robustesse (même lorsqu'un des composants n'est plus en état de remplir son rôle, le groupe peut continuer à exécuter sa tâche). De plus, le groupe n'a besoin que d'une supervision légère puisque le comportement global est émergent. Par ailleurs, lorsque les conditions du problème sont modifiées dans des propositions raisonnables, elles permettent de trouver de nouvelles solutions qui ne s'écartent pas trop des anciennes. Cette qualité est précieuse dans les problèmes réels où l'utilisateur qui doit appliquer une nouvelle solution suite à un changement dans l'environnement est ravi de ne pas avoir à chambouler toute son organisation antérieure.

Dans la vie artificielle, de nombreux chercheurs s'intéressent au fonctionnement et à l'apparitition de l'intelligence collective et utilisent l'informatique et la modélisation mathématique comme outils de prospection. Dans le cas des algorithmes fourmis, le lien avec la source d'inspiration biologique est plus évident encore et le mécanisme par dépôt de phéromone s'est popularisé en informatique des réseaux, de l'optimisation combinatoire et en data mining.

L'objet du chapitre suivant (voir le chapitre 2) est consacré à la classification, dans la première partie, nous essaierons donc de donner une définition à ce terme et détaillerons les différents types de classification. Nous verrons notamment que la classification se divise généralement en deux sous problèmes distincts : la classification supervisée et la classification non supervisée. On énumère quelques algorithmes dits traditionnels pour chaque type de classification puis on présente, dans la deuxième partie du chapitre, un certain nombre de travaux que l'on peut rassembler sous la notion assez large de fourmis artificielles.

CHAPITRE 2

CLASSIFICATION NON SUPERVISÉE

Sommaire

2.1 Introduction 39
2.2 La classification 40
2.2.1 Contexte de la classification 40
2.2.2 Définition du problème 41
2.2.3 Taxonomie des méthodes de classification ou clustering . . 41
2.2.4 Métrique et distance 42
2.2.5 Types de données en classification 43
2.3 Méthodes non hiérarchiques 43
2.3.1 Méthode K-Means 44
2.4 Méthodes hiérarchiques 46
2.4.1 Classification Hiérarchique Ascendante 47
2.4.2 Classification Hiérarchique Descendante 50
2.5 Classification incrémentale et flux de données 50
2.5.1 L'algorithme BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) 50
2.5.2 L'algorithme CURE (Clustering Using REpresentatives) . . 51
2.5.3 L'algorithme COBWEB 52
2.5.4 L'algorithme CluStream 53
2.5.5 L'algorithme DenStream 53
2.5.6 L'algorithme D-Stream 53
2.5.7 L'algorithme ClusTree 54
2.6 Les méthodes à base de grille 54
2.6.1 Les cartes auto-organisatrices (Self Organizing Maps) de Kohonen 54

2.7 Approches biomimétiques pour la classification 55
2.7.1 Fourmis artificielles 56
2.7.2 Autres approches biomimétiques : les algorithmes d'optimisation 56
2.7.3 Les algorithmes de classification automatique par fourmis artificielles 60
2.8 Méthodes de clusetring basées sur les graphes 72
2.8.1 Les graphes de voisinage et les méthodes de visualisation des graphes 73
2.8.2 CHAMELEON 81
2.8.3 Propagation d'Affinité 82
2.8.4 NG + CHL : Méthode à deux niveaux basée sur les graphes 83
2.8.5 Méthodes biomimétiques et les graphes 84
2.9 Conclusion 86

Résumé

Nous présentons dans ce chapitre un tour d'horizon des algorithmes et méthodes de classification non supervisée ainsi que de certaines méthodes de construction de graphes utilisées en apprentissage. Nous définissons le problème de la classification de données (section 2.2) et les concepts qui s'y rapportent tels que la classification conceptuelle, les approches de partionnement (K-Means et similaires) et les approches non-hiérarchiques (section 2.3). Dans la section 2.4, nous présentons des méthodes hiérarchiques (CAH, CDH).
Nous décrivons les différentes méthodes incrémentales existantes pour résoudre le problème de classification des flux de données (section 2.5) qui nous seront utiles dans la suite de ce document. Nous détaillons les approches biomimétiques pour la classification (section 2.6) telles que les algorithmes d'optimisation, les fourmis artificielles (pour lesquelles nous portons un intérêt plus marqué) et les algorithmes de classification automatique par les fourmis artificielles (AntTree, AntClass, AntClust,etc).
Nous présentons quelques algorithmes de classification basés sur les graphes et qui pourront être reliés à nos travaux comme la construction de graphes par des méthodes biomimétiques et la classification incrémentale avec des graphes (section 2.7).
Nous concluons en montrant qu'il n'existe pas encore de méthodes hiérarchiques utilisant des fourmis artificielles et que les méthodes classiques ne sont pas systématiquement les mieux adaptées pour notre problème (section 2.8).

2.1 Introduction

Le data mining offre une très grande variété de techniques et d'algorithmes de fouille de données. Ces algorithmes ont des origines diverses. Certains sont issus des statistiques (régression etc.), d'autres de l'intelligence artificielle (réseaux de neurones, arbres de décision, etc.), certains encore s'inspirent de la théorie de l'évolution (algorithmes génétiques, etc.) ou de l'éthologie (colonies d'abeilles, colonies de fourmis, etc.). Cette combinaison de technologies facilite la résolution, la compréhension, la modélisation et l'anticipation des problèmes. Ces dernières années, les besoins d'analyse de données et en particulier de classification ont augmenté significativement. En effet, de plus en plus de domaines scientifiques nécessitent de catégoriser leurs données dans un but descriptif ou décisionnel. L'envie d'organiser pour simplifier à progressivement évoluer vers l'ambition de classer pour comprendre et, pourquoi pas, pour prédire. Cette évolution a conduit à dégager des stratégies de classification.

La classification est une tâche appliquée dans la vie courante. Elle est utilisée pour expliquer les nouveaux phénomènes rencontrés en les comparant avec des concepts et des phénomènes connus et en essayant de rapprocher les caractéristiques le plus possible. C'est un sujet de recherche actif qui se place au coeur du data mining, ce qui justifie pleinement l'intérêt qui lui est porté. Ce chapitre est consacré à la classification, nous essaierons donc de donner une définition à ce terme et détaillerons les différents types de classification. Nous verrons notamment que la classification se divise généralement en deux sous problèmes distincts : la classification supervisée, appelée également analyse discriminante, et la classification non supervisée, dénommée aussi classification automatique. On énumère quelques algorithmes dits traditionnels pour chaque type de classification. Puis nous allons nous intéresser aux modèles qui copient les comportements sociaux des animaux c'est-à-dire l'intelligence en essaim qui sera étudiée sur l'espèce des fourmis dans cet état de l'art.

2.2 La classification

2.2.1 Contexte de la classification

Labroche [Labroche 2003] définit la classification comme étant 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 classifications.

Christophe Charles [Charles 2004] définit la classification automatique qui cherche à trouver une partition de l'espace de départ tel que les données appartenant à un même groupe soient plus similaire entre elles qu'avec les données issues d'un autre groupe.

D'autres définissent le clustering comme un processus de classification d'objets en groupes ou amas dont les membres sont, d'une certaine manière, similaires. Un cluster ou un groupe est donc un ensemble d'objets qui sont «similaires» entre eux et «différents» des objets appartenant aux autres groupes.

La classification de données ou «clustering» en anglais vise à partir d'un ensemble de données (individus ou objets), de regrouper ces données en sous-ensembles (groupes, classes ou clusters) d'une manière pertinente. La pertinence émanant du jugement d'un expert du domaine ou encore selon le degré d'homogénéité des groupes du point de vue des caractéristiques servant à décrire les données qui les composent. Le but de la classification, ou typologie, encore appelée taxinomie ou taxonomie, est de classer et regrouper les ensembles d'objets en sous-ensembles homogènes.

2.2.2 Définition du problème

L'objectif du clustering est d'organiser un ensemble de n objets en k clusters (groupes) tels que les objets appartenant au même cluster soient plus semblables les uns aux autres que les objets dans des différents clusters. Le clustering est l'un des outils les plus populaires pour l'exploration et l'organisation de données ; il a été largement utilisé dans presque toutes les disciplines scientifiques. Compte tenu de la croissance exponentielle de la production des données (estimées à plus de 35 trillions de gigaoctets d'ici l'an 2020), le clustering suscite un intérêt renouvelé et utilisé dans des applications telles que les réseaux sociaux, la recherche d'images, recherche sur le Web et l'analyse de l'expression des gènes. Par ailleurs, la variété des données (binaires, continues, mixtes, etc) ouvre de nouvelles perspectives et de nouvelles méthodes du clustering. Nous avons trouvé donc nécessaire d'étudier dans cette thèse l'intelligence en essaim pour le clustering. Nous proposons un algorithme bio-inspiré en se basant sur les graphes.

2.2.3 Taxonomie des méthodes de classification ou clustering

L'apprentissage automatique correspond au domaine se consacrant au développement d'algorithmes permettant à une machine d'apprendre à partir d'un ensemble de données, d'y extraire des concepts et patrons caractérisant ces données. Les algorithmes issus de ce domaine sont maintenant utilisés dans de nombreuses disciplines telles que la bioinformatique, la recherche d'information et le forage de données, etc.
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 (chez les fourmis ou les abeilles, etc.), nous effectuons une classification, et ceci de manière tout à fait naturelle.

La figure 2.1 présente les différentes variantes que l'on peut trouver parmi les méthodes de classification [Jain 1988] qui peuvent être organisées en fonction de leurs types :

- La classification non exclusive représente les méthodes de classification qui intégrent un objet dans plusieurs classes suivant un degré d'appartenance. Au contraire, la classification exclusive trie les objets de façon à ce qu'ils n'appartiennent qu'à une unique classe ;
- La classification supervisée (classification en anglais) représente les méthodes qui apprennent, à partir d'un échantillon de données déjà classées, à classer des données du même type que l'échantillon. A l'inverse, la classification non supervisée (clustering en anglais) peut passer par une phase d'apprentissage, mais le jeu de données utilisé pour cette phase n'est pas déjà classé ;
- La classification hiérarchique permet de construire une séquence de parti-

tions imbriquées, c'est à dire qu'une classe a des sous-classes, et que ces sous-classes contiennent elles aussi des sous-classes, ainsi de suite jusqu'à ce qu'un seuil soit atteint. A l'opposé, un partitionnement ne construit qu'une seule partition de données.

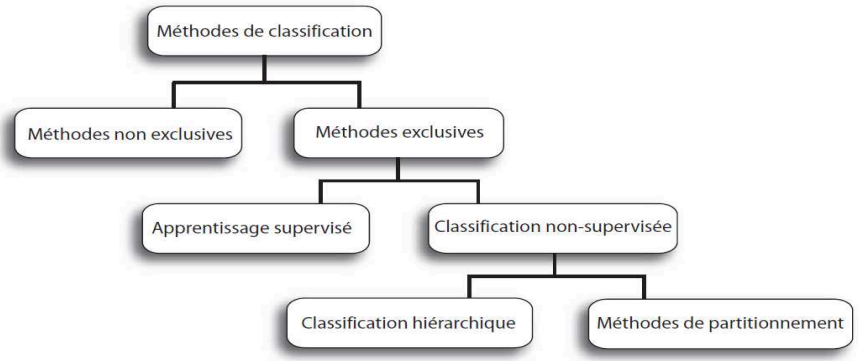


FIGURE 2.1 – Hiérarchie des méthodes de classification selon Jain et Dubes [Jain 1988].

2.2.4 Métrique et distance

La classification non supervisée se base sur le calcul d'une distance comme une mesure de similarité. Elle nécessite la résolution de ce problème : il faut définir une mesure de distance qui permet d'évaluer le niveau de similitude entre les objets à classifier. La mesure de distance la plus répandue est la distance euclidienne, mais il est possible d'utiliser d'autres distances.
L'objectif de l'analyse de données par partitionnement est de «séparer» un ensemble E composé de n objets en m partitions.
Les objets d'une même partition ont une forte similitude entre eux, et peu de similitude avec les objets des autres partitions. Pour mesurer cette similarité ou dissimilarité entre objets, une notion de distance, selon la nature des données, est nécessaire. Si ces objets sont exprimés par des variables numériques tel que l'âge, le nombre de sujets, ..., des distances telles que la distance Euclidienne, la distance de Manhattan, la distance de Chebyshev, la distance de Minkowski, ..., sont utilisées de préférence. Cependant, pour représenter de simples distances entre variables de même catégorie comme les couleurs, les familles d'animaux, le sexe, ..., on se tournera vers la distance de Jaccard ou de Hamming par exemple [Ralambondrainy 1995].

2.2.5 Types de données en classification

En classification, les données peuvent se présenter sous différentes formes :

- Données quantitatives : Elles fournissent une information enrichie et approfondie ; elles reposent sur quelques individus ou quelques cas. Elles sont utiles quand on cherche à expliquer le comment et le pourquoi (Exemples : salaires, montants des achats, âge, ...) ;
- Données qualitatives (catégorielles) : Elles proviennent le plus souvent d'études de petite taille. Elles se reposent sur l'expérience d'un groupe très restreint d'individus. Citons à ce propos l'exemple du groupe sanguin associé à une personne (O, A, B, AB) ou bien le niveau d'études d'une personne (BAC, BAC+4, ingénieur, docteur). Lorsque ces variables prennent seulement deux modalités on parle alors des variables binaires ;
- Données textuelles : Elles regroupent les textes non codés écrits en langage naturel.

2.3 Méthodes non hiérarchiques

La classification non hiérarchique ou partitionnement, aboutit à la décomposition de l'ensemble de tous les individus en m ensembles disjoints ou classes d'équivalence ; le nombre m de classes est fixé.
Etant donné un ensemble d'objets décrits par un nombre fixe d'attributs, l'objectif d'une tâche de classification non supervisée [Gordon 1996], [Kaufman L. 1990] consiste à proposer une partition des objets en k sous-ensembles où le paramètre k est le nombre de regroupements attendus par l'utilisateur.

Une variation de cette tâche est de ne pas utiliser le nombre attendu de regroupements comme une donnée du problème. Dans ce cas, l'algorithme construit plusieurs partitions candidates et choisit la meilleure. Dans le cas du clustering par partition, plusieurs méthodes se distinguent fortement [Candillier 2006] :

- **Le clustering statistique** est basé sur l'hypothèse que les données ont été générées en suivant une certaine loi de distribution, le but étant alors de trouver les paramètres de cette distribution, ainsi que les paramètres cachés déterminant l'appartenance des objets aux différentes composantes de cette loi ;
- **Le clustering basé sur les K-moyennes** est une méthode très souvent utilisée, que l'on détaille ensuite ;
- **Le clustering stochastique** consiste à parcourir l'espace des partitions possibles selon certaines heuristiques, et à sélectionner celle qui optimise un critère donné ;

– **Le clustering basé sur la densité** a pour but d'identifier dans l'espace les zones de forte densité entourées par des zones de faible densité pour la fonction des clusters ;
– **Le clustering basé sur les grilles** utilise une grille pour partitionner l'espace de description des objets en différentes cellules, puis identifie les ensembles de cellules denses connectées pour former les clusters ;
– **Le clustering basé sur les graphes** consiste à former le graphe connectant les objets entre eux et dont la somme des valeurs des arcs, correspond aux distances entre les objets, est minimale, puis à supprimer les arcs de valeurs maximales pour former les clusters ;

2.3.1 Méthode K-Means

Parmi les algorithmes de partitionnement que nous allons développer dans cette section est la méthode des k-Means. Contrairement aux méthodes hiérarchiques largement basées sur les mesures de similarités, les algorithmes de k-Means nécessitent des mesures de distance pour déterminer la distance qui sépare les individus à classer. Cette méthode est encore appelée algorithme des centres mobiles. Ce type d'algorithme, où la classe est représentée par son centre de gravité, a été étudié par plusieurs auteurs. L'algorithme k-Means mis au point par MacQueen en 1967 [MacQueen 1967] est l'un des algorithmes de clustering les plus connus. Il est basé sur la méthode des centroides (ou centres de gravité).

Ce type d'algorithme, où la classe est représentée par son centre de gravité, a été étudié par plusieurs auteurs, à savoir (Bonner, 1964 [Bonner 1964], Celeux et al., 1989 [Celeux 1989]). Le principe de cette méthode est le suivant :

• **Itération 1** : tantque l'inertie intraclasse ne s'est pas stabilisée faire ;
• **Itération 2** : générer une nouvelle partition A' en affectant chaque objet à la classe dont le centre est le plus proche ;
• **Itération 3** : calculer les centres de gravité des classes de la nouvelle partition A' ;
• **Itération 4** : A $\leftarrow$ A' ;
• **Itération 5** : fintanque ;
• **Itération 6** : retourner A.

on peut montrer que d'une itération à l'autre (i.e. d'une partition à l'autre), l'inertie intraclasse I_W décroit ce qui entraine la convergence de l'algorithme [Jain 1988]. L'algorithme k-Means (figure 2.2) assez simple à utiliser et assez efficace (très rapide) de classification non hiérarchique est l'algorithme des centres mobiles. Il se propose de prendre au hasard un centre pour chaque classe au début de l'algo-

rithme, puis de parcourir tous les points et de les affecter à la classe dont il est le plus proche du centre. A chaque itération, les centres des classes sont recalculés, puis on recommence jusqu'à arriver à des classes stables. Il présente quelques inconvénients également : il faut déterminer le nombre de classes que l'on veut au départ, ce qui est assez difficile, et l'algorithme est très dépendant des centres choisis au départ (aléatoirement dans la version classique). De plus, il ne détecte bien que les classes de formes sphériques (il donne une mauvaise classification sur les classes ellipsoïdes allongées par exemple, etc.).
Cette figure ci-après illustre le fonctionnement générique de la méthode des K-Means.

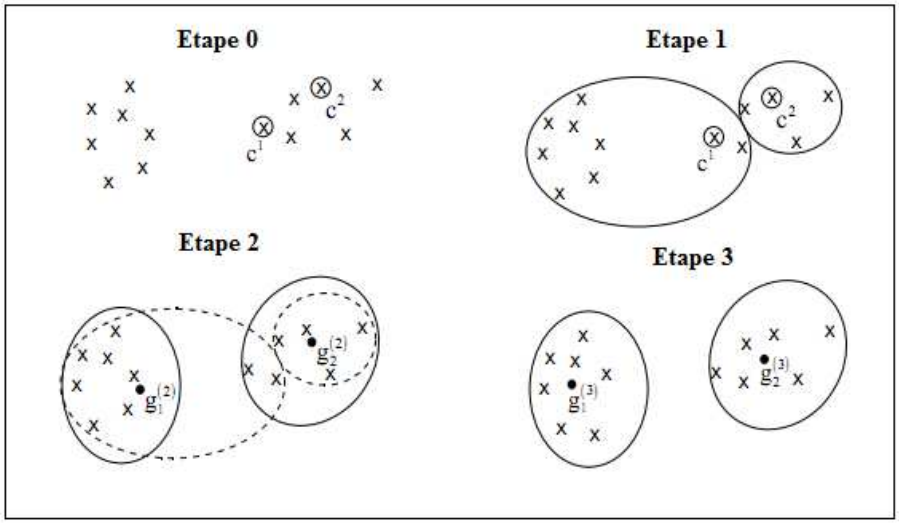


FIGURE 2.2 – Fonctionnement générique de l'algorithme K-Means.

- **Etape0** : choix des centres c_1 et c_2
- **Etape1**
  - Construction de classes autour des centres c_1 et c_2.
  - Classe 1 : points plus proches de c_1 que de c_2.
  - Classe 2 : points plus proches de c_2 que de c_1.
- **Etape2**
  - Calcul des centres de gravité des classes formées à l'étape g_1 et g_2.
  - Définition de nouvelles classes autour des centres de gravité.
- **Etape3**
  - Calcul des centres de gravité des classes formées à l'étape2.
  - Nouvelles définitions autour de ces centres.

  $\rightarrow$ Stabilité (critère d'arrêt) : si aucun point n'a changé de groupe, les groupes sont stables.

Au titre de ses forces se trouve le principe des méthodes de k-Means qui consiste à calculer le centre de chaque classe. Cela permet de réduire la complexité de ces algorithmes car c'est désormais à ce centre que seront comparés tous les autres individus des autres classes ou nouveaux individus entrants afin de déterminer leur classe d'appartenance. Dans une application de recherche d'information utilisant les k-Means comme méthode de classification, les requêtes et les nouveaux documents seront comparés uniquement au vecteur du centre, et non plus à chaque paire de documents. C'est en ce sens que les algorithmes de k-Means sont rapides et qu'ils sont utilisés dans de nombreuses applications. Le calcul d'un individu centre permet également au résultat de cette méthode d'être appliqué à des taches de prédiction. Ainsi, cette fixation a priori du nombre de classes à obtenir constitue également son inconvénient majeur car, de fait, le point de départ (le nombre de k fixé) détermine le point d'arrivée (le nombre de classes obtenues). Dans son principe même, la méthode des k-Means s'appuie sur une hypothèse forte, celle qui consiste à supposer que l'utilisateur connait a priori le nombre optimal de classes à former. Hartigan et Wong [Hartigan 1979] ont fourni une mise en oeuvre simple et efficace de k-Means, lorsque la distance métrique pour le regroupement est euclidien. Cependant, cet algorithme peut-être pas approprié pour d'autres métriques de similarité de groupement.

2.4 Méthodes hiérarchiques

Pour les auteurs dans [Everitt 1993], [Atlan 2006] et [Jain 1988] la classification hiérarchique consiste à effectuer une suite de regroupements en classes de moins en moins fines en agrégeant à chaque étape les objets ou les groupes d'objets les plus proches. Elle fournit ainsi un ensemble de partitions de l'ensemble d'objets [Celeux 1989]. Cette approche utilise la notion de distance, qui permet de refléter l'homogénéité ou l'hétérogénéité des classes. Ainsi, on considère qu'un élément appartient à une classe s'il est plus proche de cette classe que de toutes les autres.
L'approche hiérarchique ne se contente pas d'une seule partition, mais crée une hiérarchie de parties qui constituent un arbre.
Une classification hiérarchique n'est autre qu'une suite de séquence de partitionnements. Etant donné un ensemble d'observations, une hiérarchie sur cet ensemble est une collection de groupes d'observations (clusters) tels que :

- L'ensemble complet des données est un cluster ;
- Chacune des observations est un cluster (singleton) ;
- Etant donnés deux clusters de la hiérarchie, ou bien ils n'ont aucune observation en commun, ou bien l'un est inclus dans l'autre (pas de chevauchement).

Pour des raisons pratiques, il convient d'imposer également que chaque cluster (excepté les singletons) est partitionné en exactement deux clusters de la hiérarchie. Une telle structure peut se représenter par un «dendogramme» (ou «arbre»). Différentes contraintes sont bien sûr imposées : chaque groupe doit être le plus ho-

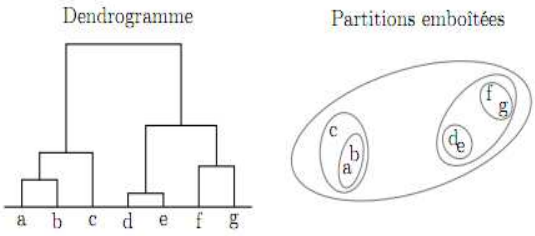


FIGURE 2.3 – Un dendogramme (ou «arbre») [Celeux 1989].

mogène possible et les groupes doivent être les plus différents possible entre eux.

Les méthodes dites hiérarchiques sont parmi les plus anciennes pour former automatiquement des classes d'objets [Hartigan 1975] et [Johnson 1967]. Ces méthodes construisent une hiérarchie de classes emboitées. Cette hiérarchie peut être ascendante (classification Ascendante Hiérarchique ou CAH) ou descendante (Classification Descendante Hiérarchique ou CDH).

2.4.1 Classification Hiérarchique Ascendante

La Classification Hiérarchique Ascendante (CHA)(ou «par agrégation») procède par fusions successives de clusters déjà existants. A chaque étape, les deux clusters qui vont fusionner sont ceux dont la «distance» est la plus faible. La question est donc de trouver une bonne définition de ce que l'on entend par la «distance» entre deux groups de points. Il existe de nombreuses définitions d'une telle distance, la plus utilisée étant la distance de Ward. La Classification Ascendante Hiérarchique (ou «CAH») considère initialement toutes les observations comme étant des clusters ne contenant qu'une seule observation (singleton), et leur distance est alors le plus souvent définie comme étant leur distance euclidienne. La première étape consiste donc à réunir dans un cluster les deux observations les plus proches. Puis la CAH continue, fusionnant à chaque étape les deux clusters les plus proches au sens de la distance choisie. Le processus s'arrête quand les deux clusters restant fusionnent dans l'unique cluster contenant toutes les observations.
La règle la plus communément utilisée pour tracer l'arbre de la hiérarchie est telle que les typologies qui ont le plus de chance d'être significatives sont obtenues simplement en traçant une ligne horizontale en travers du dendogramme, et en retenant

dans la typologie les clusters terminaux qui sont juste au-dessus de cette ligne. En changeant l'hauteur de la ligne, on change le nombre de clusters retenus, et on dispose ainsi d'un moyen simple pour faire varier la granularité de la typologie finale. Cette possibilité n'existe pas dans K-Means, où le nombre de clusters doit être fixé à l'avance.

- **Itération 1** : k éléments sont à classer, les k éléments forment chacun une classe ;
- **Itération 2** : l'algorithme calcule les distances entre chacun des k éléments et regroupe les deux plus proches. On obtient ainsi une première partition en (k-1) classes ;
- **Itération 3** : l'algorithme recalcule les distances entre le nouvel élément et les éléments restants, puis regroupe à nouveau les deux éléments les plus proches. On obtient alors une deuxième partition à (k-2) classes ;
- **Itération n** : le processus est réitéré jusqu'à la dernière partition qui constitue une seule grande classe regroupant l'ensemble des éléments.

La suite des partitions obtenues est généralement représentée sous la forme d'un arbre de classification. Les distances (ou niveaux d'agrégation) représentent le niveau auquel ont lieu les regroupements. Plus on monte dans l'arbre, plus la distance est élevée et plus les classes sont hétérogènes.

Le problème des méthodes hiérarchiques ascendantes est de trouver l'ultra métrique (distance entre distance) la plus proche de la métrique utilisée pour les individus. L'utilisateur choisit un niveau de classification, on parle alors du niveau de coupure de dendrogramme. Pour considérer la valeur des liens entre les classes déjà obtenues et les objets à classer, plusieurs de critères d'agrégation sont utilisées : le lien minimal, le lien moyen, le lien complet, le saut de ward.

L'agrégation par le lien minimal ou lien simple (single link) regroupe en priorité les deux classes les plus proches, à savoir celles dont la dissimilarité est la plus petite. Ce critère d'agrégation, bien qu'intuitif, provoque rapidement un effet de chaine sur une matrice de grande de taille. En effet, il suffit qu'entre deux classes il existe un seul lien fort pour qu'elles soient regroupées. L'effet de chaine conduit à des classes trop longues, peu denses et pouvant aussi avoir des contenus hétérogènes. Par conséquent, le lien minimal est déconseillé de manière quasiment consensuelle dans la littérature [Atlan 2006], [Everitt 1993], sauf lorsque l'application visée nécessite de créer ce type de classes.

Le lien moyen (average link) réunit deux objets selon la moyenne arithmétique des liens entre ces deux objets. Cette technique, la plus utilisée dans la littérature, donne des résultats plus satisfaisants en termes de cohérence interne des classes. La technique d'agrégation par lien complet (complet link), appelée également la méthode de diamètre ou la méthode de saut maximum, est l'opposée du lien mi-

nimal. Elle regroupe deux classes en fonction de la distance la plus éloignée entre deux éléments quelconques de ces classes.

Le lien complet conduit à la formation de petites classes denses et mieux délimitées [Bellot 2000], [Wong 2007a], mais il n'est pas adapté à des données. Il est aussi recommandé dans des cas où les éléments d'une même classe sont assez éloignés dans l'espace. Cette technique donne plus de poids aux éléments atypiques (non centraux) de deux classes.

Le saut de ward regroupe des classes en fonction de la moyenne des carrés des distances entre points. Le saut de ward nécessite la définition d'un espace métrique euclidien. Cette technique conduirait à la formation de classes homogènes et à une hiérarchie symétrique [Bellot 2000]. Le résultat d'une CAH n'est pas une partition de l'ensemble des individus. C'est une hiérarchie de classes telles que :

- Toute classe est non vide ;
- Tout individu appartient à une (et même plusieurs) classes ;
- Deux classes distinctes sont disjointes, ou vérifient une relation d'inclusion (l'une d'elles est incluse dans l'autre) ;
- Toute classe est la réunion des classes qui sont incluses dans elle.

Dès 1993, Allen a utilisé la CAH afin de regrouper les documents similaires dans une base encyclopédique [Allen 1993]. L'interface proposée à l'utilisateur se compose alors du dendogramme résultat de la CAH dans lequel il est possible de se déplacer en visualisant le document en cours. Cutting et al. [Cutting 1992] ont utilisé une méthode hybride mêlant la méthode des K-Means pour son efficacité et la CAH pour sa qualité afin de définir une interface permettant à l'utilisateur de naviguer dans un ensemble de documents en éparpillant et en regroupant des ensembles de documents pour un grand volume de données.

Il existe de très nombreuses implantations des algorithmes de classification hiérarchique. Certaines implantations tentent de remédier à des handicaps constatés des algorithmes de base ou de les adapter à des types de données spécifiques [Eisen 1998] et [Yeung 2003]. Ainsi Eisen et al. [Eisen 1998] ont développé la plateforme Cluster dotée d'un outil de visualisation (TreeView) et incluant plusieurs techniques de classification, dont la Classification Ascendente Hiérarchique. Une faiblesse des algorithmes hiérarchiques est leur complexité linéaire. Le calcul de similarité doit être effectué entre toutes les paires d'objets dans la phase d'agrégation, et l'introduction de nouveaux objets implique de recommencer ce calcul. Le nombre de comparaisons à faire croit avec le nombre d'objets à comparer. La complexité de ces algorithmes est donc égale au nombre d'objets élevé au carré (O (n_2)).

2.4.2 Classification Hiérarchique Descendante

La Classification Hiérarchique Descendante (ou «par division») procède de façon inverse. Elle considère l'ensemble des données comme un gros cluster unique, et le scinde en deux clusters «descendants». La scission s'opère de façon à ce que la distance entre les deux descendants soit la plus grande possible, de façon à créer deux clusters bien séparés. Cette procédure est ensuite appliquée à chacun des descendants (procédure récursive) jusqu'à ce qu'il ne reste plus que des clusters ne contenant qu'une seule observation (singletons). La même représentation en dendogramme, et la même procédure de définition d'une typologie à partir d'un dendogramme utilisées par l'approche «Ascendante» sont utilisables par approche «Descendante». Pour réaliser la subdivision, il faut souvent faire une classification hiérarchique ascendante pour savoir quelle est la meilleure façon de séparer les points.

2.5 Classification incrémentale et flux de données

Depuis ces dernières années plusieurs méthodes utilisant le principe de l'approche incrémentale ont vu le jour. Qu'elles soient conceptuelles comme dans le système COBWEB [Fisher 1991] ou non conceptuelles comme l'algorithme BIRCH [Zhang 1996] ou l'algorithme CURE [Guha 1998]. On appelle souvent cette opération classement ou catégorisation. Ces méthodes fonctionnent toutes à partir d'une mesure de distance qui doit être fournie par l'utilisateur. Les méthodes plus récentes qui n'exigent pas de distance préalable sont AutoClass (un classifieur bayésien), COBWEB un système dédié aux descripteurs symboliques et les cartes auto-organisatrices («Self Organizing Maps») de Kohonen. On décrit aussi quelques méthodes de clustering de flux basées sur la densité comme DenStream, CluStream et D-Stream.

2.5.1 L'algorithme BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies)

BIRCH est un algorithme qui travaille efficacement sur de gros jeux de données. L'idée principale de ce dernier est la classification qui est effectuée sur un résumé vraiment compact des données, au lieu des données originales. C'est pourquoi il peut traiter un grand volume de données en utilisant une mémoire limitée. Il est incrémental, i.e. il a besoin d'un seul balayage du jeu de données. Il essaie de minimiser le coût d'entrée/sortie en organisant les données traitées en une structure d'arbre équilibré avec une taille limitée. La communauté du domaine de classification trouve que BIRCH est l'un des meilleurs algorithmes qui peuvent traiter de

gros jeux de données.

Initialement, BIRCH a été créé pour traiter des données numériques, une extension a été proposée pour traiter des données de type catégorie. Tout d'abord, nous en parlerons au niveau algorithmique sans aborder en détail la mesure (le calcul) utilisée dans l'algorithme. BIRCH utilise la notion Cluster Feature (CF en abréviation) qui est utilisée pour représenter une classe ou une sous-classe. C'est un vecteur dont les informations stockées dépendent du type de données qu'on traitera mais dans tous cas, il faut respecter deux exigences :

- La similarité (ou la distance) entre un couple de classes est facile à calculer à l'aide des CFs ;
- Le CF de chaque classe est facilement mis à jour lors d'une insertion d'un nouveau membre ou d'une fusion des classes. Dans BIRCH, on utilise l'addition comme opération pour la mise à jour du CF d'une classe.

Le principal avantage de l'algorithme BIRCH [Zhang 1996] par rapport aux méthodes traditionnelles est sa rapidité d'exécution. Cette méthode repose principalement sur la représentation du jeu de données initial sous forme d'un arbre d'attributs de cluster plus connu en anglais sous le terme de Clustering Features Tree (CF-Tree). L'arbre est construit de manière incrémentale, ce qui permet de traiter de grands volumes de données. Les données manipulées par BIRCH sont seulement de type numérique. Ainsi chaque objet est représenté par son vecteur caractéristique.
Une fois les données représentées sous forme arborescente, le regroupement est réalisé par la méthode de classification ascendante hiérarchique. Une limite importante de l'algorithme BIRCH est son impossibilité de faire face à des classes de tailles variées, de forme non sphériques, ou reliées par un ensemble de points (problème d'effet de chaîne). En effet, la majorité des algorithmes de classification apportent de meilleurs résultats lorsque les classes du jeu de données initial sont bien distinctes.

2.5.2 L'algorithme CURE (Clustering Using REpresentatives)

Il a été proposé par Guha et al. [Guha 1998]. Il est une approche hybride : utiliser tous les points de données et ainsi un centroid pour former les classes : Il représente une classe par un nombre constant de points de représentation appelés les représentants. Pour obtenir ces points, tout d'abord, on choisit les points qui s'éparpillent bien autour d'une classe. Ensuite on les fait rétrécir vers le centroide de la classe par une fraction quelconque. Ces points éparpillés après le rétrécissement seront utilisés comme les représentants de la classe.
L'idée principale de l'algorithme CURE pour éviter les limitations de l'algorithme BIRCH est de sélectionner un ensemble de points représentatifs des classes et de

les rapprocher du centre de la classe grâce à un facteur α (ce qui permettra aussi d'éviter les points isolés). Ensuite, CURE utilise un algorithme de classification ascendante hiérarchique, et pour décider quelles classes fusionner, on choisit les deux classes dont les points représentatifs sont les plus proches. Pour manipuler de grandes bases de données, CURE utilise un échantillonnage aléatoire sur la base de données initiale, les données restantes sont ensuite affectées aux classes dont les centres sont les plus proches. Cette manière de faire permet à CURE d'être plus rapide que BIRCH dans la phase de pré-classification.
Cette méthode emploi un nouvel algorithme de classification ascendante hiérarchique qui adopte un compromis entre les points extrêmes et les points centraux de chaque classe. Ce compromis est basé sur le principe d'utilisation d'un ensemble de points représentatifs pour chaque classe. Au début, ces points sont choisis de manière aléatoire et déplacés ensuite vers le centre de la classe grâce à un facteur de rapprochement α. Ainsi, dans le processus de classification, deux classes sont fusionnées si leurs points représentatifs sont les plus proches. Par rapport aux méthodes basées sur les centroïdes, ce type d'approche a l'avantage de ne pas exclure les points éloignés.

CURE est donc moins sensible aux points isolés que les autres méthodes. Ainsi, le fait de représenter chaque classe par un ensemble de points permet à CURE de mieux identifier les classes de forme non sphérique. Une fois la classification sur l'échantillon aléatoire terminée, le reste de la base de données peut être classé. Pour cela, au lieu d'utiliser les centroïdes comme représentant des k classes générées on choisira un ensemble de points représentatifs pour chaque classe. A partir de là, chaque point est affecté à la classe dont les points représentatifs sont les plus proches.

2.5.3 L'algorithme COBWEB

COBWEB est une méthode de classification conceptuelle, le principe de cette méthode est de construire de manière incrémentale une hiérarchie de concepts où chaque noeud est un concept probabiliste décrit par la probabilité P(Ck) d'appartenir à ce concept et la probabilité conditionnelle P(A_j = v_{ij} |Ck) qu'un objet de la classe Ck prenne la valeur v_{ij} sur l'attribut A_j.

Cependant la construction de la hiérarchie est guidée par une mesure heuristique d'évaluation de la qualité d'une partition : the category utility function (fonction d'utilité d'une catégorie) qui utilise des principes généraux cités dans les méthodes de classification non supervisée. En effet, cette fonction est un compromis entre la similarité intra-classe et la dissimilarité inter-classe.
COBWEB construit des hiérarchies de classes en utilisant une «mesure d'utilité». Étant donné la classification courante et un nouvel exemple, COBWEB mesure quelle classe devrait accueillir le nouvel exemple, ou si une nouvelle classe de-

vrait être créée, en calculant l'utilité de chacune des actions. Ceci est tout à fait semblable à certains algorithmes classiques, avec cette différence que la mesure de distance est fournie avec l'algorithme (c'est la mesure d'utilité), et il n'y a pas besoin de spécifier combien de classes on désire obtenir.
Un tel «COBWEB» continu a été utilisé avec succès sur un large ensemble de données visuelles [Ketterlin 1994]. Ceci indique clairement que COBWEB n'est pas seulement valable pour de petits ensembles de données quoique que ce soit un algorithme très lent. Son incrémentalité, en effet, permet d'ajouter continuellement des exemples à la hiérarchie.

2.5.4 L'algorithme CluStream

CluStream [Aggarwal 2003] est un algorithme robuste basé sur la classification automatique des données numériques. L'idée consiste à construire un résumé du flux de données sous la forme d'un ensemble de micro-classes évolutives dont des clichés sont mémorisés régulièrement. Pour développer cet algorithme, Aggarwal s'est inspiré de l'algorithme BIRCH [Zhang 1996] en utilisant la structure des CFVs (Cluster Feature Vector). Cet algorithme essaie de faire la classification sur le flux de données entier plutôt que regarder le flux de données comme un processus de changement au cours du temps. Il fournit un framework pour la classification le flux de données évolutives dans lequel il y a une grande variété de fonctionnalités telles que la création de macro classes sur différents horizons temporels et l'analyse de l'évolution du flux de données.

2.5.5 L'algorithme DenStream

L'algorithme DenStream [Cao 2006] repose sur un résumé du flux par des micro-clusters mais différencie les clusters potentiels des points aberrants potentiels. Les micro-clusters sont classés par rapport à leur poids. Il s'agit d'un algorithme du classification automatique des données numériques. Cet algorithme hérite du fonctionnement de CluStream. Il fournit un résumé sous forme dun ensemble de micro-classes évolutives. Contrairement à CluStream, DenStream définit deux types de micro-classes. Des micro-classes pour les évènements atypiques nommées o-micro-classe, et des micro-classes pour le reste des évènements nommées p-micro-class.

2.5.6 L'algorithme D-Stream

D-Stream [Tu 2009] est un l'algorithme à base du densité qui est présenté par une grille dans laquel les points de données sont dressés sur la carte correspondante et les grilles sont groupées à base de leur densité. Les grilles sont pondérées par les

récents points. Cette méthode fait correspondre à chaque bloc de données d'entrée en une grille et il calcule la densité de la grille. Cet algorithme adopte une technique de décomposition de densité pour capturer les changements dynamiques d'un flux de données (voir la figure 2.4).

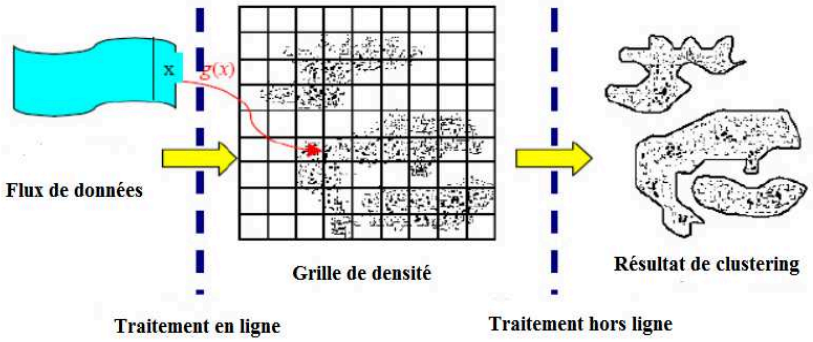


FIGURE 2.4 – Vue générale de l'algorithme D-Stream.

2.5.7 L'algorithme ClusTree

Cette méthode [Kranen 2011] repose généralement sur un algorithme sans paramètre qui ne réalise qu'une seule passe sur les données initiales, avec l'utilisation de la mémoire limitée. Il s'adapte automatiquement à la vitesse du flux de données. ClusTree présente des solutions pour gérer des flux très rapides grâce à des mécanismes d'agrégation et proposer des stratégies de descente nouvelles qui améliorent le résultat de regroupement sur les flux lentes. Les expériences montrent que cette approche est capable de traiter une multitude de caractéristiques de flux pour le clustering de flux à tout moment d'une manière précise et évolutive. ClusTree n'est malheureusement applicable qu'à des données vectorielles.

2.6 Les méthodes à base de grille

2.6.1 Les cartes auto-organisatrices (Self Organizing Maps) de Kohonen

Inspirés du fonctionnement de cerveau humain, les cartes auto-organisatrices de Kohonen [Kohonen 2001] constituent une méthode d'apprentissage non supervisé, contrairement à la plupart de méthodes issues du modèle neuronal. Les cartes auto-organisatrices constituent un cas particulier des réseaux de neurones, dont

l'objectif est de représenter des données d'une grande dimension dans un espace à deux dimensions tout en préservant leurs propriétés topologiques et les distances entre elles. Les cartes auto-organisatrices se présentent comme une grille à deux couches, de forme rectangulaire ou hexagonale. La première couche est constituée de stimulus d'entrée (la donnée d'entrée à classifier ou ses propriétés), et la deuxième, des noeuds (données déjà disponibles). Chaque noeud de la grille est relié au vecteur (poids) du stimulus d'entrée.
Une fois que la phase d'apprentissage est terminée, la carte se stabilise (est organisée). Chaque stimulus d'entrée est placé à l'endroit du noeud gagnant. Etant donné que l'objectif d'une méthode de classification est de réduire la dimension de départ, il fut utilisé une grille contenant moins de noeuds que le nombre d'objets à classifier. La taille de la grille a une importance capitale pour la qualité des résultats. Cela revient de fait à choisir le nombre de k classes à former, problème aussi difficile que celui de trouver le nombre optimal de classes pour une méthode de K-Means. Moins il y a de noeuds, plus faible sera la résolution, mais plus la taille des objets assignés à chaque noeuds grandit, ce qui rend difficile l'exploration du contenu des noeuds. A l'inverse, il est déconseillé de choisir un nombre trop grand de noeuds car le temps d'apprentissage n'en sera que plus long. Dans leur application à la classification de données bibliographiques en astronomie, Poinçot et al. [Poincot 1998] proposent un compromis à ce problème, qui consiste à créer une grille secondaire, reliée à la carte principale, pour explorer le contenu des noeuds trop densément peuplés. Ils recommencent alors un nouveau cycle d'apprentissage sur la carte secondaire, laquelle comporte un nombre plus limité de noeud.

2.7 Approches biomimétiques pour la classification

L'intelligence artificielle en essaim recouvre un ensemble d'algorithmes utilisés en informatique qui ont la particularité de faire interagir de multiples processus élémentaires de façon décentralisée, c'est-à-dire sans contrôleur global, afin de résoudre des problèmes. Elle est née de la modélisation mathématique et informatique des phénomènes biologiques rencontrés en éthologie. L'informatique s'est en effet inspirée des études faites sur l'intelligence en essaim biologique pour produire des algorithmes novateurs. L'exemple de la conception des algorithmes fourmis est révélateur de cette bio-inspiration. En retour les algorithmes conçus en informatique sont aussi des outils d'analyse et de simulation pour les biologistes.

Le développement de méthodes d'analyse dynamique de l'information, comme les méthodes de clustering et les méthodes de détection de nouveauté, devient une préoccupation centrale dans un grand nombre d'applications dont le but principal est de traiter de larges volumes d'information variant au cours du temps. Ces ap-

plications se rapportent à des domaines très variés et hautement stratégiques, tels que l'exploration du Web et l'analyse dynamique des données pour les nouvelles générations de moteurs de recherche et la recherche d'information, l'analyse du comportement des utilisateurs et les systèmes de recommandation, la veille technologique et scientifique, ou encore, l'analyse de l'information génomique en bioinformatique. Il existe, en outre, des algorithmes ceux que nous venons de citer, qui découlent de la biologie s'appuyant sur des modèles d'organisation du travail chez les fourmis où les fourmis résolvant de nombreux problèmes de manière très efficace dans leur environnement, il doit être possible de transposer un certain nombre de ces mécanismes pour la résolution de problèmes informatiques, parmi ces problèmes on spécifie des algorithmes de classification suggérés par les essaims d'insectes plus précisément les insectes fourmis. Dans ce chapitre nous nous intéressons aux algorithmes biomimétiques inspirés des comportements des fourmis pour résoudre le problème de classification.

2.7.1 Fourmis artificielles

Les fourmis artificielles sont issues de l'observation de la nature : les sociétés animales, telles que les colonies de fourmis, ont développé des comportements collectifs d'une efficacité et d'une robustesse fascinantes. La motivation d'imiter la nature a été forte et les fourmis artificielles ont connu un développement rapide dans la communauté des chercheurs, principalement confrontés à des problèmes d'optimisation. De nombreux travaux en vie artificielle utilisent les fourmis comme support. Certaines études s'intéressent à l'apparition de phénomènes complexes, similaires aux phénomènes collectifs naturels. Le premier algorithme de colonies de fourmis a été conçu pour optimiser le problème du voyageur de commerce. Ce problème consiste à chercher le trajet le plus court reliant N villes données, chaque ville ne devant être visitée qu'une seule fois. Les fourmis sont des insectes sociaux extrêmement répandus et adaptés à leur environnement [Holldobler 1990]. Leurs mécanismes de communication sont souvent élaborés et ont inspiré les scientifiques principalement pour la résolution de problèmes d'optimisation combinatoire tel que le classique voyageur de commerce par exemple [STÜTZLE 1999].

2.7.2 Autres approches biomimétiques : les algorithmes d'optimisation

Les algorithmes biomimétiques sont des modélisations inspirées du principe qu'un groupe d'entités plutôt simples et obéissant à des règles locales de coordination, sont capables d'engendrer des comportements globaux beaucoup plus complexes.
Ces modélisations caractérisent ce que l'on dénomme «l'intelligence collective»

[Monmarché 2000] ou encore «l'intelligence en essaims»
[Bonabeau 1999b] qui regroupe de nombreux algorithmes biomimétiques. Ces algorithmes ont déjà prouvé leur efficacité face aux problèmes d'optimisation [Dréo 2003], [Guha 1998].

2.7.2.1 L'algorithme ACO

Les algorithmes à base de fourmis dont le principe général a donné naissance à la méta-heuristique d'optimisation par colonies de fourmis ACO [Dorigo 1992], [Dorigo 2006] et [Dorigo 2004] construisent également des configurations de façon incrémentale, mais choisissent le prochain composant en fonction d'une règle de transition probabiliste qui dépend généralement de deux facteurs : un facteur heuristique similaire à celui utilisé par les algorithmes gloutons aléatoires et un facteur phéromonal représente une fonction des «traces de phéromones» déposées autour du composant. Ces traces de phéromones représentent l'expérience passée de la colonie de fourmis concernant le choix de ce composant et sont régulièrement mises-à-jour en fonction des dernières configurations calculées.

L'optimisation par colonies de fourmis (ACO pour Ant Colony Optimisation) est une méthode évolutive inspirée du comportement des fourmis à la recherche de nourriture. Cet algorithme a été proposé par Dorigo en 1992. Il est connu que les fourmis sont capables de déterminer le chemin le plus court entre leur nid et une source de nourriture. Ceci est possible grâce à la phéromone qui est une substance que les fourmis déposent sur le sol lorsqu'elles se déplacent. Lorsqu'une fourmi doit choisir entre deux directions, elle choisit avec une plus grande probabilité celle comportant une plus forte concentration de phéromone.

C'est ce processus coopératif qu'ACO tente d'imiter. Les fourmis réelles ont inspiré les chercheurs en informatique dans de nombreux domaines. Cela se justifie particulièrement quand on connait la richesse comportementale de ces animaux. L'un des modèles les plus connus a été introduit par Colorni et al. [Colorni 1991], initialement dans le cadre du problème du voyageur de commerce, le principe consiste à «lancer» des fourmis, et à les laisser élaborer pas à pas la solution, en allant d'une ville à l'autre. C'est donc un algorithme qui repose sur la construction progressive de solutions. Afin de ne pas revenir sur ses pas, une fourmi tient à jour une liste Tabou, qui contient la liste des villes déjà visitées. 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). Cette approche a été depuis largement développée et appliquée à de nombreux problèmes d'optimisation combinatoire et numérique. Soit A un ensemble de k fourmis :

Pour construire un trajet, chaque fourmi se construit une route en choisissant

les villes selon une règle de transition aléatoire très particulière : Si $P_{ij}^k(t)$ est la probabilité qu'à l'itération t la fourmi k choisisse d'aller de la ville i à la ville j, alors on a :

$$P_{ij}^k(t) = \frac{\tau_{ij}(t)^\alpha \cdot \eta_{ij}^\beta}{\sum_{l \in J_i^k} \tau_{il}(t)^\alpha \cdot \eta_{il}^\beta} \text{ Si } j \in J_i^k$$

$$P_{ij}^k(t) = 0 \text{ Si } j \notin J_i^k$$

Où :
$\tau_{ij}(t)$ désigne le taux de phéromone sur la route ij à l'itération t.
η_{ij} désigne l'inverse de la distance séparant les villes i et j.
α et β sont deux paramètres contrôlant respectivement l'influence du taux de phéromone sur le trajet ij, et l'influence de la distance sur le trajet ij.

En d'autres termes, plus il y a de phéromone sur le trajet reliant deux villes, plus la probabilité est grande que la fourmi emprunte ce trajet. Mais ceci est contrebalancé par la longueur du trajet. α et β permettent de régler l'effet de ces paramètres. Lorsque toutes les fourmis ont construit une solution, les taux de phéromone τ sur les routes en fonction des trajets effectivement empruntés par les fourmis seront mis à jour, selon la formule :

$\Delta\tau_{ij}^k(t) = \frac{Q}{L^k}(t)$ Si le trajet (i, j) est dans la tournée de la fourmi k à l'itération t.

Q est une constante, et $L^k(t)$ est la longueur totale de la tournée de la fourmi k. On constate donc que plus la route suivie par la fourmi a été courte, plus grande est donc la quantité de phéromone laissée derrière elle. Pour éviter que des chemins ne se forment pas vite, et ne convergent pas rapidement vers des optima locaux : à la fin de chaque itération de l'algorithme, les phéromones déposées aux itérations précédentes par les fourmis s'évaporent de $\rho\tau_{ij}(t)$. Et à la fin de l'itération, on a la somme des phéromones qui ne se sont pas évaporées et de celles qui viennent d'être déposées :

$$\tau_{ij}(t+1) = (1-\rho) \cdot \tau_{ij}(t) + \sum_{k=1}^{m} \Delta\tau_{ij}^k(t)$$

Avec m est le nombre de fourmis utilisées pour l'itération t et ρ esy un paramètre de réglage.

Il existe plusieurs variantes de l'algorithme ACO on peut citer parmi eux :

– **Ant System (AS)** : l'algorithme Ant System est le premier algorithme d'optimisation par colonies de fourmis. Il a été proposé en 1996 par Dorigo, Maniezzo et Colorni dans [Dorigo 1996] ;

- **Ant System et élitisme** : une première variante du «Système de Fourmis» a été proposée dans [Dorigo 1996] 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 le but d'accroître la probabilité des autres fourmis d'explorer la solution la plus prometteuse ;
- **Max-Min Ant System (MMAS)** : cet algorithme est une autre amélioration de AS, proposée en 2000 par Stülze et Hoos [Stützle 1997]. La différence avec AS réside dans le fait que :
  1. Seule la meilleure fourmi dépose des traces de phéromones.
  2. Les valeurs minimales et maximales de phéromones sont explicitement fixées.
- **Ant-Q** : dans cette variante de l'AS, la règle de mise à jour locale est inspirée du «Q-learnings» [Gambardella 1995]. Cependant, aucune amélioration par rapport à l'algorithme AS n'a pu être démontrée. Cet algorithme n'est d'ailleurs, d'après les auteurs, qu'une pré-version du «Ant Colony System» ;
- **ACS et 3-opt** : cette variante est une hybridation entre l'ACS et une recherche locale de type 3-opt
  Le tableau ci-dessous [Boulmerka 2009] représente une liste non-exhaustive des algorithmes d'optimisation par colonies de fourmis :

Algorithme	Auteurs	Références	Années
ANT SYSTEM (AS)	DORIGO ET AL.	[Colorni 1991], [Dorigo 1997]	1991
ELITIST AS	DORIGO ET AL.	[Dorigo 1996]	1992
ANT-Q	GAMBARDELLA et AL.	[Gambardella 1995]	1995
ANT COLONY SYSTEM (ACS)	DORIGO et AL.	[Dorigo 1997]	1996
MAX-MIN ANT SYSTEM (MMAS)	STUTZLE et HOOS	[Stützle 1997]	1996
RANK-BASED AS	BULLNHEIMER ET AL.	[Bullnheimer 1999]	1997
ANTS	MANIEZZO	[Franks 1992]	1999
BWAS	CORDON ET AL.	[Cordon 2000]	2000
HYPER-CUBE ACO	BLUM et AL.	[Blum 2004]	2004
INVARIANCE ACO	BIRATTARI ET AL.	[Birattari 2007]	2007

TABLE 2.1 – Liste des algorithmes d'optimisation par colonies de fourmis. [Boulmerka 2009]

2.7.2.2 L'algorithme API

L'algorithme API, qui est quant à lui inspiré par le comportement de recrutement d'une fourmi primitive [Monmarché 2000a]. Les auteurs préconisent ce recrutement particulier pour faire progresser la population vers l'optimum, en choisissant le meilleur point parmi ceux évalués par les fourmis.

Dans tous les algorithmes évoqués jusqu'ici, le terme «colonies de fourmis» implique l'utilisation de la stigmergie comme processus d'échange d'information. Cependant, il existe un algorithme adapté au cas continu développé par Monmarché et al. [Monmarché 2000] qui s'inspire du comportement de fourmis primitives de l'espèce «Pachycondyla Apicalis». Cet algorithme peut être résumé ainsi :

- un nid est positionné aléatoirement sur l'espace de recherche ;
- ensuite, des fourmis sont distribuées aléatoirement dans l'espace. Ces fourmis vont alors explorer localement leur «site de recherche» en évaluant plusieurs points dans un périmètre donné (voir figure 2.5) ;
- chaque fourmi mémorise le meilleur point trouvé. Si, lors de l'exploration de son site de recherche, elle trouve un meilleur point, alors elle reviendra sur ce site, sinon, après un certain nombre d'explorations, elle choisira un autre site ;
- une fois les explorations des sites de recherche terminées, des fourmis tirées au hasard comparent deux à deux leurs meilleurs résultats, puis mémorisent le meilleur des deux sites de chasse ;
- le nid est finalement réinitialisé sur le meilleur point trouvé après un temps donné, la mémoire des sites des fourmis est remise à zéro, et l'algorithme effectue une nouvelle itération.

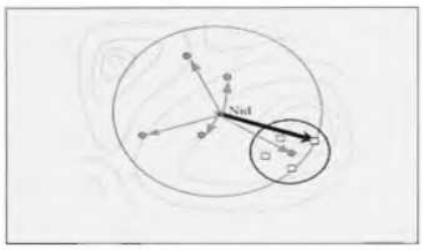


FIGURE 2.5 – Fonctionnement de l'algorithme API : Les fourmis (cercles pleins) explorent des sites de recherche (petits carrés) dans un périmètre (grand cercle) autour du nid [Dréo 2004].

2.7.3 Les algorithmes de classification automatique par fourmis artificielles

Nous verrons dans les sections suivantes de notre état de l'art, un éventail des différentes méthodes à bases de fourmis artificielles, l'intelligence en essaim. Ce sont des approches qui ont pour principe de reproduire certains comportements observés chez les fourmis réelles en réponse à des tâches complexes

dificilement réalisables pour un individu aussi simple qu'une fourmi et de les appliquer principalement à des problèmes de classification.

2.7.3.1 L'algorithme LF

Dans la nature, certaines espèces de fourmis sont capables d'organiser spacialement divers éléments du couvain : les oeufs, les larves et les nymphes. Le modèle basé sur les travaux de Deneubourg et ses collègues [Deneubourg 1991], [Franks 1992]. Pour rassembler en tas un ensemble d'éléments (d'objets) de même type, les probabilités de ramasser un objets p_p et de le déposer p_d ont été explicitées [Lumer 1994b] quand une fourmi ne transporte aucun élément, sa probabilité d'en ramasser un, rencontré sur son chemin, est donnée par :

Formule 1 : $p_p = \left(\frac{K_1}{K_1+f}\right)^2$
où K_1 est une constante positive et f correspond à la proportion d'éléments perçus dans le voisinage de la fourmi. Quand il y a peu d'objets dans le voisinage de l'objet convoité par la fourmi, f $\ll K_1$ ce qui signifie que p_p est proche de 1 et l'objet a beaucoup de chance d'être ramassé. Inversement, quand le voisinage est dense en éléments, f $\gg K_1$ et alors p_p est proche de 0. Quand une fourmi chargée d'un objet se déplace, sa probabilité de déposer l'objet est donnée par :
Formule 2 : $p_d = \left(\frac{f}{K_2+f}\right)^2$
où K_2 est une constante positive. L'évaluation de f est proposée pour une implantation en robotique : f correspond au nombre d'objets rencontrés durant les T derniers déplacements divisé par le nombre maximum d'objets qui auraient pu être rencontrés.

Le pas qui sépare le tri d'objets de la classification a ensuite été franchi par Lumer et Faieta en 1992 avec l'algorithme LF puis amélioré en 1996 [Monmarché 2000a]. Cet algorithme a été, depuis, étendu à d'autres applications comme le partitionnement de graphes (voir [Kuntz 1997])ou la classification de sessions sur des sites Web (voir [Abraham 2003]). Une autre extension de l'algorithme LF a été proposée en 2002 dans [Ramos 2002] avec l'algorithme ACluster pour la classification de documents. A la différence de LF, les auteurs suppriment l'utilisation de vitesse ou encore de mémoire pour les fourmis. Ces derniers n'apportent, selon les auteurs, que des difficultés de paramétrage de l'algorithme mais surtout s'écarte un peu trop de la réalité biologique représentée. Ils utilisent par conséquent, un modèle basé sur un dépôt de phéromones par les fourmis lors de leurs déplacements. Ainsi les densités de phéromones sont plus importantes dans les zones de la

grille comportant des objets et très faible dans les endroits où il y en a peu [Labroche 2003].

• **Principe de fonctionnement**

Lumer et Faieta [Lumer 1994a] ont proposé un algorithme utilisant une mesure de dissimilarité entre les objets (sous la forme d'une distance euclidienne). Les objets qui, rappelons-le, correspondent à des points d'un espace numérique à dimensions sont plongés dans un espace discret de dimension moindre (typiquement de dimension 2). Cet espace discret s'apparente alors à une grille G dont chaque case peut contenir un objet. Les agents se déplacent sur G et perçoivent une région R de $S \times S$ cases dans leur voisinage. La figure 2.6 donne un exemple de grille avec une fourmi (représentée par $\times$) et son périmètre de détection (en trait épais). Les objets sont représentés par des carrés dont l'intérieur (invisible pour la fourmi) représente la classe d'origine.

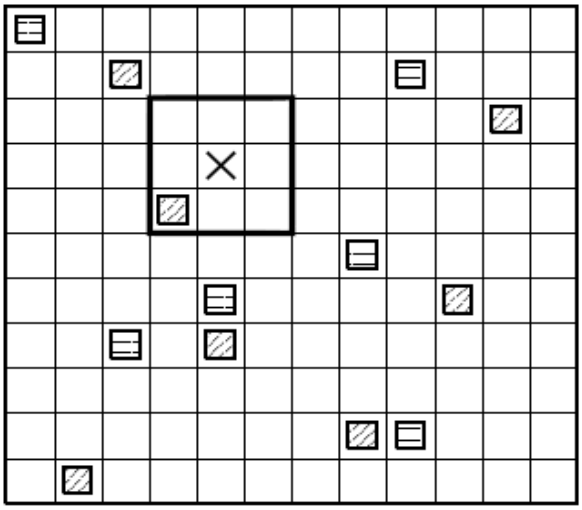


FIGURE 2.6 – Grille utilisée dans E.Lumer et B.Faieta [Monmarché 2000] : La fourmi est représentée par $\times$ et son périmètre de détection par un trait épais. Les objets sont représentés par des carrés dont l'intérieur («invisible» pour la fourmi) représente la classe d'origine.

2.7.3.2 L'algorithme AntClass

Une des premières études relatives au domaine des fourmis artificielles a été menée par [Deneubourg 1991] où une population d'agents-fourmis se déplacent aléatoirement sur une grille à deux dimensions et sont capables de déplacer des objets dans le but de les rassembler. Cette méthode à été étendue par [Lumer 1994a] sur des objets simples puis par [Kuntz 1997] où

un problème réel est abordé dans le but de résoudre efficacement un problème d'optimisation. Le travail assez récent a été mené par Monmarché [Monmarché 2000] et ses collègues dont l'intérêt principal est d'introduire les principes de classification par des fourmis dans des problèmes d'analyse de données. Ce travail a abouti à un nouvel algorithme de classification non supervisée appelé «AntClass» couple l'algorithme LF avec l'algorithme des centres mobiles (K-Means) afin de résoudre les inconvénients inhérents à la classification par les fourmis, par exemple pour accélérer la convergence en éliminant les erreurs «évidentes» de classification.

Cet algorithme peut être considéré comme une synthèse et une extension de tous les travaux déjà existants dans ce domaine. L'algorithme AntClass a été testé sur plusieurs types de données : des données artificielles, des données réelles issues de la base de données «Machine Learning Repository» [Taillard 1998] et des problèmes réels en coopération avec l'industrie. Les résultats obtenus ont été jugés pertinents, positifs et plus performants que les résultats obtenus par les algorithmes classiques tels que, le K-Means et l'Isodata [Monmarché 2000a].

AntClass introduit une classification hiérarchique dans la population de fourmis artificielles qui seront aussi capables de transporter des tas d'objets. 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. 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 (voir la figure 2.7).

Dans AntClass, les fourmis se déplacent aléatoirement, il est donc possible qu'elles ne passent pas sur les bons tas au bon moment. Pour limiter ce phénomène, nous obligeons les fourmis à parcourir tous les clusters avant de créer un nouveau cluster. Cependant, pour éviter que les fourmis se dirigent au même endroit en même temps, ce parcours est aléatoire : à chaque visite d'un cluster, la fourmi le marque comme visité. De plus, en procédant ainsi le nombre de clusters à un instant donné sera borné par le nombre de cluster réel. Autrement dit, on cherche à limiter le nombre de tas pouvant être fusionnés.

•**Principe de fonctionnement**

Au départ, les fourmis $a_1, a_2, ..., a_n$ sont réparties aléatoirement sur la grille

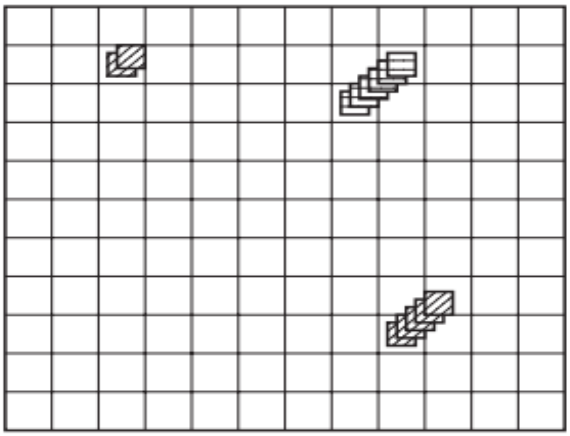


FIGURE 2.7 – AntClass permet la construction de tas d'objets sur la grille [Monmarché 2000].

en vérifiant qu'une case ne peut contenir qu'une seule fourmi. A chaque itération de l'algorithme, chaque fourmi a_i se déplace aléatoirement sur la grille à une vitesse v(a_i). La direction choisie par une fourmi dépend de sa direction précédente. Chaque fourmi est dotée d'une capacité de transport c(a_i). Si la fourmi a_i ne transporte pas d'objets et qu'elle se trouve sur une case contenant un objet ou un tas d'objets T_j, elle a une probabilité p_p de ramasser un objet, qui dépend d'un paramètre de contrôle de la forme de la densité de la probabilité et du nombre d'objet dans le tas. Si la fourmi transporte un objet, et qu'elle se trouve sur une case contenant un ou plusieurs objets, sa probabilité p_d de déposer l'objet o_i sur le tas T_j dépend d'un paramètre réel positif permettant de contrôler la forme de la densité.
Si la capacité de la fourmi est supérieure à 1 et qu'elle transporte plusieurs objets, la probabilité de déposer le tas T_i qu'elle transporte sur le tas T_j est calculée de la même façon que pour un objet unique en remplaçant o_i par le centre de gravité g_i des objets transportés.
S'il y a trop de fourmis il se peut que tous les objets soient transportés, ce qui n'offre plus la possibilité de dépôt pour les fourmis (absence de tas). Dans le cas où la capacité des fourmis est égale à 1, il suffit de choisir un nombre de fourmis inférieur aux nombres d'objets. Dans le cas contraire (capacité des fourmis supérieur à 1), on dote les fourmis d'une patience p(a_i). Quand une fourmi effectue un nombre de déplacement supérieur à p(a_i) sans réussir à déposer les objets qu'elle transporte, elle les dépose sur la case où elle se trouve si elle est vide ou l'une de son voisinage dans le cas contraire.

– **Déplacement d'une fourmi de la liste vers les agents cluster** Lorsqu'une fourmi arrive sur la liste, elle ramasse aléatoirement un objet dans la liste.

Puis elle se dirige vers les agents cluster pour essayer de déposer l'objet dans un des clusters. Une fourmi qui va sur la liste ne doit pas porter d'objet.

– **Déplacement d'une fourmi des agents cluster vers la liste** Lorsqu'une fourmi arrive sur la liste des agents cluster, elle choisit au hasard un agent cluster et vérifie si elle peut y déposer l'objet qu'elle transporte. Si elle ne peut pas, elle en choisit un autre, et ainsi de suite. Si elle n'a pas déposé son objet dans un des agents cluster, elle crée un nouvel agent cluster contenant cet objet avant de retourner sur la liste.

AntClass utilise les fourmis en deux étapes : une première étape, où les fourmis travaillent sur les objets, puis, après un passage par l'algorithme K-Means, les fourmis travaillent sur les tas d'objets, pour finir par un deuxième passage par l'algorithme K-Means. Cette stratégie permet d'obtenir, en quelques milliers d'itérations, des résultats beaucoup plus stables et proches du partitionnement souhaité.

2.7.3.3 L'algorithme AntTree

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 chaine, c'est-à-dire qu'elles s'accrochent entre elles (elles s'auto-assemblent) puis courbent et rapprochent les feuilles. 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. L'algorithme principal de construction d'arbres par des fourmis artificielles est [Maziz 2004].

Dans [Anderson 2002], les auteurs ont regroupé plusieurs phénomènes d'auto-assemblage que l'on peut observer chez les animaux, en particulier les insectes sociaux. Le modèle réel d'auto-assemblage qui est une inspiration pour développer ces algorithmes de fourmis est basé sur l'étude réalisée sur les fourmis Linepithema humile et Oecophylla Longinoda par [Sauwens 2000] et [Lioni 2000] durant leur thèse de doctorat. Azzag et al. [Azzag 2004] ont développé un modèle de règles comportementales pour des fourmis artificielles pour la classification non supervisée hiérarchique. Ces travaux présentent un exemple d'utilisation de l'algorithme AntTree pour la construction des sites portails pour la recherche de documents sur le web. Un site portail

est vu comme une classification hiérarchique d'un ensemble de documents en catégories et sous catégories. Ils proposent pour cela une modélisation nouvelle inspirée des fourmis réelles qui n'a jusqu'à présent encore jamais été appliquée à la résolution de problèmes en informatique.

Il s'agit de modéliser la manière dont les fourmis forment des structures vivantes [Lioni 2001], [Theraulaz 2001] et d'utiliser ce comportement pour organiser les données à regrouper selon un arbre qui se construit de manière distribuée. Ce phénomène d'auto-assemblage chez les fourmis réelles peut être résumé de la manière suivante : à partir d'un point de support fixe sur lequel sont situées initialement les fourmis, ces dernières vont s'accrocher successivement au support, puis aux fourmis connectées au support, et ainsi de suite jusqu'à ce que, par exemple, une chaîne de passage soit construite entre deux points. Les fourmis se déplacent sur la structure vivante et s'accrochent sur celle-ci aux endroits les plus opportuns en fonction du but à atteindre et de la configuration locale de la structure.
Les sous-arbres apparaissent au premier niveau de l'arbre, juste sous le support, constituent le résultat de la classification trouvée par l'algorithme. Chaque sous-arbre correspond à une classe constituée de toutes les données présentes dans ce sous-arbre (voir la figure 2.8).

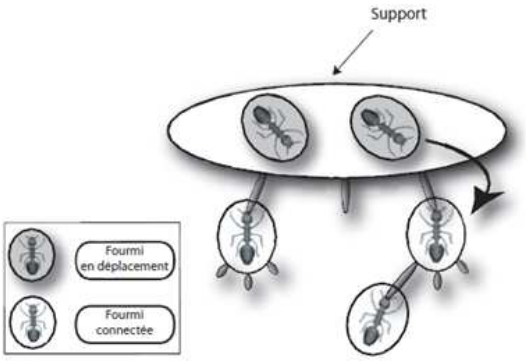


FIGURE 2.8 – Construction de l'arbre par des fourmis : principe général. Les fourmis qui sont en déplacement sont représentées en gris et les fourmis connectées en blanc [Azzag 2007].

2.7.3.4 L'algorithme TTA

W.Wong et al. [Wong 2007b] s'intéressent au problème de la construction d'ontologie. Ce problème peut s'aborder avec des techniques de classification non supervisée pour constituer des groupes de termes se rapportant

à un même concept. Comme les corpus d'apprentissage de ces ontologies sont en permanente évolution, les auteurs cherchent à profiter des capacités d'adaptation des fourmis en utilisant des ressources d'information du web (Google et Wikipedia). L'algorithme proposé, appelé TTA (Tree Traversing Ant) construit un arbre binaire contenant les termes en partant d'un noeud initial (la racine) qui, au démarrage, contient tous les termes à classer.

Les fourmis se saisissent d'un terme dans la racine et le déplacent vers l'un des deux noeuds fils, en fonction d'une mesure de similarité entre deux termes. Cette similarité est construite en interrogeant le moteur de recherche Google. Ensuite, l'arbre construit est corrigé en utilisant la structure de graphe de l'encyclopédie en ligne Wikipedia. La structure utilisée pour le regroupement par le TTA est un arbre dynamique. L'arbre dynamique commence avec un noeud racine r_0 composé de tous les termes T = $T_1, ..., T_n$, et continuera à se ramifier à de nouveaux sous-noeuds que nécessaire.

Les TTAs ne sont pas différents de la fourmi standard en termes de nombre d'objets qu'ils peuvent effectuer à un moment donné et de l'incapacité pour la communication directe, les phéromones peuvent être nécessaires comme un mécanisme de rétroaction positive de son environnement. TTAs sont trop dotés de la capacité de mémoire à court terme pour se rappeler les similitudes et les distances acquises grâce à ses sens. L'algorithme de regroupement proposé commence le traitement au niveau du noeud racine qui se compose de tous les termes de r_0 = $T_1, ..., T_n$. Chaque terme peut être considéré comme un élément dans noeud. Le TTA au hasard choisir un terme, et procéder à sens sa ressemblance avec tous les autres termes de ce même noeud. Le TTA répète ce pour tous les termes n jusqu'à ce que la similitude de toutes les paires possibles de termes ont été mémorisé.

Le TTA a été conçu comme un essai de faire fusionner les forces de méthodes à base de fourmi standard avec de certains avantages de méthodes de clustering conventionnelles pour la classification des termes dans l'apprentissage d'ontologie. En plus, une seconde passent pour raffiner les résultats produits par le TTA utilisant Distance Google Normalisée (NGD). Le TTA emploiera une nouvelle mesure de Wikipedia (n˚W) pour quantifier la distance entre deux termes basés sur la transliaison d'articles Wikipedia.

2.7.3.5 L'algorithme $AntCO^2$

Antoine Dutot et al. [Dutot 2005] ont proposé un algorithme nommé $AntCO^2$ est une approche dans laquelle une population d'entités informatiques simples

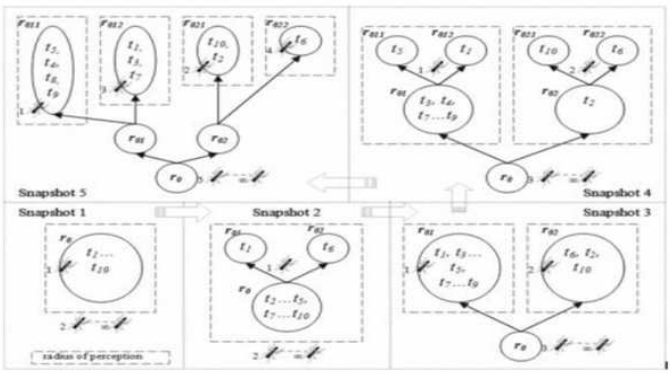


FIGURE 2.9 – Exemple du fonctionnement de l'algorithme TTA [Wong 2007b].

et réactives (par opposition à des agents proactifs) imitant une partie du comportement de fourmis naturelles, coopèrent et entrent en compétition pour la colonisation de zone de forte interaction d'un graphe dynamique.
L'un des rôles majeurs de l'algorithme $AntCO^2$ que nous décrivons est de détecter des organisations dans un graphe en fonction d'un critère de communication. L'algorithme $AntCO^2$ signifie «Ant Colored colonies», l'$AntCO^2$ est un modèle dans lequel non seulement des fourmis numériques collaborent pour effectuer une tâche comme pour de nombreuses autres approches fourmi, mais aussi dans lequel ces mêmes individus entrent en compétition. La majorité des algorithmes basés sur les fourmis ne reprennent qu'un de ces aspects : la collaboration. Cet algorithme capable de tirer profit des organisations au sein d'applications comme des simulations d'écosystème, par exemple. Dans $AntCO^2$, il existe plusieurs colonies de fourmis, chacune d'une couleur unique, représentant une ressource de calcul donnée, qui se «confrontent» afin de coloniser des zones d'un environnement représentant l'application à distribuer. Ainsi il y a collaboration au sein d'une colonie, mais compétition entre les colonies. C'est la collaboration qui va permettre la détection d'organisations au sein des entités de l'application, et donc des groupes à distribuer, et c'est la compétition qui va permettre une distribution de la charge de ces groupes sur différentes ressources de calcul.

Les fourmis colorées d'$AntCO^2$ déposent des phéromones elles-aussi colorées. Ainsi chaque fourmi laisse dans son environnement non seulement une quantité donnée de phéromone, mais aussi un message qualitatif indiquant sa colonie. Les fourmis utilisent un graphe dynamique pondéré non orienté coloré. $AntCO^2$ est une méthode itérative. À chaque itération, les fourmis traversent un arc. La phéromone perçue par les fourmis est modulée par la présence d'autres phéromones sur l'arc. C'est cette partie qui permet le com-

portement de répulsion face aux phéromones de colonies «adverses». Les fourmis d'une couleur c sont bien attirées par les phéromones de couleur c, et plus la quantité de cette phéromone est élevée plus elles sont attirées, mais en outre cette attraction est modulée par le fait qu'il peut y avoir bien plus ou bien moins de phéromones d'autres couleurs. Ainsi à chaque itération, une fourmi k de couleur c placée sur un noeud u choisit d'emprunter un arc plutôt qu'un autre en fonction des phéromones de sa couleur et des autres couleurs présentes sur ces arcs.

2.7.3.6 L'algorithme ANTPART

Cette méthode est inspirée du comportement de fourragement (recherche de nourriture) d'une espèce de fourmis tropicale vivant au Mexique appelée «Pachycondyla Apicalis». Le principe étant simple, chaque fourmi chasse en solitaire et tente de couvrir uniformément un espace donné autour de leur nid par des recherches locales sur des zones de chasses [Maziz 2004], [Admane 2006].

• **Principe de fonctionnement**

Nous proposons de définir quelques caractéristiques relatives aux fourmis et à l'approche AntPart avant d'aborder le principe général.

– Fourmi Ouvrière : la fourmi ouvrière est la fourmi de base qui effectue la recherche des proies (nourriture) dans les sites de chasse. C'est-à-dire que c'est la fourmi qui effectue l'affectation des données ;
– Fourmi Maître : la fourmi maître est la fourmi qui s'éloigne du nid pour augmenter les chances de trouver de la nourriture améliorant ainsi la partition obtenue ;
– Mémoire Fourmi : chaque fourmi a la capacité de mémoriser les sites de chasse qu'elle a visités. C'est-à-dire que chaque fourmi possède une mémoire contenant les données qu'elle a classé ;
– Mémoire Colonie : c'est une mémoire commune à toutes les fourmis, elle contient les différentes classes créées ;
– Similarité : deux données transportées par deux fourmis sont similaires si leurs caractéristiques sont suffisamment identiques ;
– Dissimilarité : deux données transportées par deux fourmis sont dissimilaires si leurs caractéristiques sont suffisamment différentes.

Initialement, tous les objets sont dans un nid. Chaque fourmi ouvrière f_i choisit (dans le nid) aléatoirement une donnée D_i à transporter pour lui chercher une classe (un site). La fourmi f_i recherche, dans sa mémoire locale, un objet assez similaire à D_i : si celui-ci est trouvé, alors affecter à sa classe l'objet D_i transporté. Dans le cas contraire, la fourmi entame la recherche

d'objets similaires à D_i dans la mémoire de la colonie. Après recherche dans la mémoire de la colonie, deux situations se présentent : soit l'objet le plus similaire est assez similaire à D_i, il faut alors affecter D_i à la classe de l'objet trouvé. Soit les deux objets ne sont pas assez similaires mais assez dissimilaires, il faut alors créer une nouvelle classe contenant D_i. Dans le cas où les deux objets ne sont ni assez similaires ni assez dissimilaires alors D_i est remis dans le nid, et ses seuils de similarité et de dissimilarité sont mis à jour.

2.7.3.7 L'algorithme AntClust

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 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 (du à l'évaporation), elles attaquent toutes 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. L'algorithme AntClust s'inspire du phénomène de reconnaissance chimique chez les fourmis. Il repose sur l'apprentissage et le partage d'une odeur coloniale commune à toutes les fourmis d'un même nid [Maziz 2004] [Labroche 2003].

L'idée générale est d'associer à une fourmi artificielle (ou plus exactement à son génome) une donnée à classer. La fourmi va ensuite essayer de trouver la colonie qui l'intègre le mieux. Une fourmi i possède les paramètres suivants :

- le label $label_i$, modélisé par une variable qui représente l'indice du nid d'appartenance de la fourmi i ;
- Le génome $Gnome_i$ représente l'objet à classer ;
- Le template $Template_i$ est un seuil d'acceptation d'un objet dans une classe ;
- L'estimateur M_i estime le nombre de fourmis ayant le même label que la fourmi i ;
- L'estimateur M_i^+ mesure l'intégration de la fourmi i dans son nid ;
- L'âge A_i, qui au départ vaut 0, est utilisé dans les calculs de mise à jour du seuil d'acceptation $Template_i$.

Les fourmis font un nombre fini de rencontres avec d'autres fourmis. Ceci leur permet d'identifier le nid (ou label) correspondant à leur génome. Pour arriver à trouver le nid adéquat, chaque fourmi se base sur son template, qu'elle a appris durant sa jeunesse et auquel sont comparés tous les labels (odeurs) des fourmis rencontrées. Par les rencontres des fourmis ayant un génome similaire, la construction progressive de la partition finale se fait. Cette partition se construit sans connaissance à priori du nombre de classes. C'est une approche non supervisée de la classification, qui est utilisée dans AntClust. L'algorithme AntClust passe par quatre étapes essentielles :

- **Etape d'apprentissage** : dans cette étape, des rencontres entre les fourmis sont effectuées durant NApp itérations dans le but d'initialiser le template des fourmis avant l'affectation. A chaque rencontre, la similarité entre deux fourmis est calculée, pour que les similarités maximales et moyennes soient modifiées. Ceci étant fait, le template des deux fourmis est initialisé ;
- **Etape d'affectation** : dans cette étape, les fourmis se rencontrent deux à deux pendant NbIter itérations. Deux fourmis s'acceptent dans un même nid si la similarité entre ces deux fourmis est supérieure aux deux templates :
  Acceptation (i, j) = (Sim(i, j) > $Template_i$) (Sim(i, j) > $Template_j$)
- **Etape de suppression** : la suppression s'effectue comme suit : On calcule, pour chaque nid, le taux d'intégration des fourmis. Ce taux est comparé à un seuil S'. Les nids dont la valeur d'intégration est inférieure à S', sont soumis à un tirage aléatoire pour être supprimés ou pas. Donc plus l'intégration des fourmis dans le nid est faible (par rapport à S'), plus il a

de chances d'être détruit ;
– **Etape de réaffectation** : chaque objet d'un des nids supprimés est affecté au nid qui lui est le plus similaire. Plusieurs améliorations peuvent être ajoutées à AntClust, pour réduire la complexité de l'algorithme ou le rendre incrémental. Il est même envisageable d'étudier le parallélisme de l'algorithme afin de répartir les différentes tâches de la classification.

A partir de ce principe nous avons développé un modèle de règles comportementales pour des fourmis artificielles qui permettent l'émergence de structures au niveau global. Dans cette thèse, nous montrons comment nous nous avons basé sur ces principes pour définir un algorithme capable de classer de manière non supervisée et non hiérarchique des données de plusieurs types.

2.8 Méthodes de clusetring basées sur les graphes

Les algorithmes basés sur la représentation sous forme de graphes sont reconnus pour les bonnes performances du clustering obtenu. En particulier les clusters peuvent être reconnus quels que soient leurs formes, contrairement à la plupart des méthodes basées sur le calcul de centres représentatifs des clusters (K-Moyennes par exemple). Cependant, le problème majeur avec ce type d'algorithmes est la complexité des calculs mis en oeuvre. En effet, le nombre d'arêtes à considérer est souvent proportionnel au carré du nombre de données, ce qui peut poser problème dans le cas de grandes bases de données.

Les méthodes à deux niveaux ont cependant l'avantage de réduire considérablement cette complexité, puisque le nombre d'arêtes est alors proportionnel au nombre de prototypes, qui est en général bien plus petit que le nombre de données. C'est une autre approche proposée dont de nombreux auteurs est de considérer les relations de distance ou de similarité entre données comme un graphe [Matula 1977], [Wu 1993], [Guha 1998], [Hartuv 1999]. La classification de données à base de graphes est un domaine riche en méthodes et algorithmes. Nous présentons dans cette section différents algorithmes récents se basant sur une représentation sous forme de graphe et qui présentent de bonnes performances.

Il existe différentes solutions afin de visualiser un graphe [Di Battista 1993b]. Cependant, il n'existe pas de méthode parfaite et chacune dépend des contraintes imposées par la nature des données du graphe et de sa structuration. Certaines solutions, tel que le framework de visualisation Tulip de [Auber 2002], réa-

lisent une visualisation de grands graphes (plusieurs milliers de noeuds à la fois). Ceci dans le but de visualiser un maximum de données (si possible le graphe initial dans son intégrité). Cependant, ces solutions ne permettent pas de mettre l'accent sur une partie précise (locale) du graphe ni d'interagir sur son ensemble. Elles produisent le plus souvent une vue statique du dessin de graphe.

La quantité d'information ne cesse de croître de jours en jours. Il est impossible pour les experts de se passer d'outils de recherche automatiques. Nous avons ainsi décrit différentes techniques de visualisation, proposées comme un état de l'art non exhaustif compte tenu du très grand nombre de travaux et présenté des méthodes d'affichage dans différents types d'applications.

Nous présentons, dans un premier temps, les différents modèles de graphes de voisinage couramment utilisés. Dans un second temps, nous proposons un tour d'horizon des méthodes de visualisation les plus utilisées en fouille de données. Ainsi, dans cette étude, l'objectif est de se focaliser sur les méthodes de dessin de graphes utilisés dans le domaine de la visualisation de données. Dans la deuxième partie, nous choisissons une technique de visualisation de données utilisée pour la visualisation des données telle que la méthode Tulip [Auber 2002] a déjà été utilisée pour notre problématique nécessitant de visualiser les relations de voisinage dans un ensemble de données, nous nous sommes naturellement intéressés aux graphes de voisinage et à la représentation visuelle de graphes.

2.8.1 Les graphes de voisinage et les méthodes de visualisation des graphes

Nous nous intéressons dans cette section aux méthodes traditionnelles de construction de graphes de voisinage les plus couramment utilisées en réponse à des problèmes divers et variés. Les graphes de voisinage ou graphes de proximité sont des structures géométriques qui utilisent le concept de voisinage pour déterminer les sommets les plus proches d'un sommet donné. Pour cela, ils se basent sur les mesures de «distances» [Toussaint 1991]). Prenons un graphe G (Ω, V) où Ω est l'ensemble des noeuds et V l'ensemble des arêtes entre les noeuds. Lorsque G est un graphe de voisinage, les noeuds représentent des données, et les arêtes représentent une relation entre les données qui indique laquelle des données sont voisins. Dans ce cas, pour un noeud donné d_i de Ω , le voisinage (d_i) représente l'ensemble de tous les noeuds qui sont connectés à d_i. Construire un graphe de voisinage consiste à utiliser une distance existante (ou de la similarité) entre les don-

nées dans le but d'établir des relations binaires entre les noeuds.

Un graphe de voisinage met en évidence, de manière cohérente, les relations de voisinage existantes au sein d'un ensemble. Le principe de construction, pour la plupart des modèles de graphes de voisinage, repose sur l'utilisation d'une contrainte mathématique qui consiste à rechercher pour chaque point d'un ensemble donné si les autres points appartiennent à son voisinage. Les méthodes de construction de graphe de voisinage utilisant une contrainte mathématique, cela consiste à construire le graphe final en partant du graphe complet. La construction s'effectue étape par étape, du graphe précédent au graphe suivant par suppression successives des arêtes ne respectant par la propriété de construction. La figure 2.10 nous présente un exemple de graphe de voisinage.

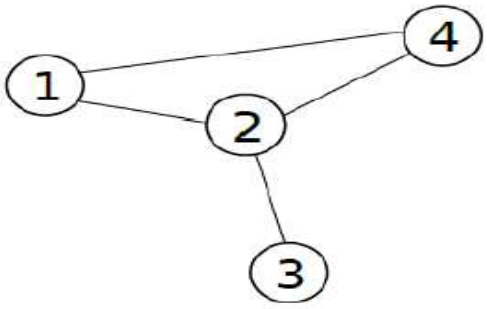


FIGURE 2.10 – Exemple d'un graphe de voisinage.

Il existe plusieurs méthodes pour construire un graphe de voisinage. Nous présentons les modèles classiques suivants : le diagramme de Voronoi, la triangulation (graphe) de Delaunay, le graphe de Gabriel, et le graphe des voisins relatifs. Tous ces modèles permettent de produire un graphe connexe G (V, E) c'est à dire qu'il existe un chemin entre tous les sommets.

2.8.1.1 Diagramme de Voronoi

Un mathémacien, du nom de Georgi Fedoseevich Voronoi [1868-1908], propose une nouvelle méthode de décomposition d'un espace métrique déterminée par les distances à un ensemble discret d'objets de l'espace, en général un ensemble discret de points. Il s'agit de réaliser une partition de cet espace en tenant compte des distances entre les éléments dun ensemble. Le plus souvent, ces objets sont les points d'un ensemble fini et discret. Chaque point devient le centre d'une région dite de Voronoi. Lun des algorithmes les

plus performants est celui de [Fortune 1997]. Dans [Edelsbrunner 1987] se trouve les informations sur la résolution de problèmes par les diagrammes de Voronoi classiques. La définition 1 précise le type de partitionnement global que l'on nomme diagramme de Voronoi qu'illustre la figure 2.11.

Définition 1. Considérons V un ensemble fini de n points dans un espace euclidien F de dimension d (remarque 2). Pour chaque point v_i de V avec i $\in$ [1...n], la cellule de Voronoi correspondante à v_i est définie de la manière suivante : elle est constituée de l'ensemble des points de l'espace F qui sont le plus proche de v_i que de tous les autres points de V. Le diagramme de Voronoi est la décomposition de l'espace F en un ensemble de cellules de Voronoi (i.e. autant de cellules de Voronoi que de sites).

Remarque 1. Une région est également appelée cellule de Voronoi. Son périmètre détermine la forme géométrique du polygone de Voronoi.

Remarque 2. Les éléments de V sont également appelés sites, centres ou germes.

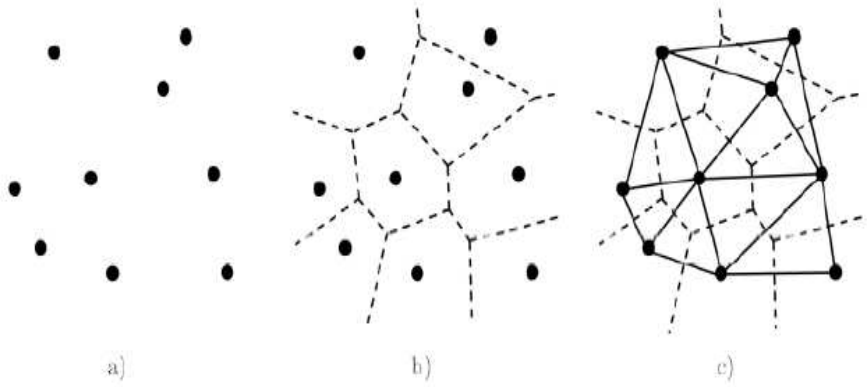


FIGURE 2.11 – Diagramme de Voronoi - a) Nuage de sites V dans un espace F de dimension 2. b) Diagramme de Voronoi correspondant à l'ensemble de sites V dans F. c) Triangulation de Delaunay ou dual du diagramme de Voronoi de la figure 2.11 b).

2.8.1.2 Triangulation de Delaunay

La triangulation de Delaunay, du nom du mathématicien Delone Boris [1890-1980] (francisé en Delaunay), est le dual géométrique du diagramme de Vo-

ronoi. Nous proposons deux définitions pour construire la triangulation de Delaunay. La définition 2 compose initialement la triangulation à partir d'un ensemble de sites tandis que la définition 3 se base sur le diagramme de Voronoi, préalablement construit, pour déterminer la triangulation de Delaunay. Pour les travaux qui sont faits sur la triangulation de Delaunay, on peut citer les travaux de [Cignoni 1998]. Pour le RNG, une extension a été proposée dans [Hacid 2007] où les données sont ajoutées de manière incrémentale. Le test de connexion est donc limité à un sous-ensemble de l'ensemble des données.

De cette manière, le temps de calcul peut être amélioré. Cependant, ce temps de calcul peut être réduit davantage. La triangulation de Delaunay [Preparata 1985] qui, en 2D, connecte tout triplet de données qui vérifie la propriété suivante : le cercle qui passe par les trois points ne contient pas d'autres données. Cette approche peut être généralisée en 3D. Elle répond à des problématiques de triangulation de domaines et possède des applications similaires à son dual [Okabe 1992]. Une définition a été proposé se base sur le diagramme de Voronoi, préalablement construit, pour déterminer la triangulation de Delaunay [Tran Dac 2004].

Définition 2. Considérons un ensemble V de n points dans un espace de dimension 2. Pour chaque triplet de points de V par lesquels passe un cercle ne contenant aucun autre point de V , il existe une unique triangulation (remarque 3). L'ensemble des triangulations obtenues constituent la triangulation de Delaunay pour l'ensemble V de sites. La figure 2.12 illustre la construction de la triangulation de Delaunay à partir du diagramme de Voronoi de la figure 2.11.b.

Remarque 3. En dimension 3, nous utiliserions, à la place de cercles, des sphères ou encore des tétraèdres.

Définition 3. Soit un ensemble V de n sites dans un espace euclidien F de dimension n. La triangulation de Delaunay est le dual du diagramme de Voronoi, pour l'ensemble de sites de V , s'il existe pour chaque paire (v_i, v_j) de cellules de Voronoi adjacentes, une arête reliant les centres v_i et v_j de ces régions.

Remarque 4. Si l'on s'en tient à la définition 3, la triangulation de Delaunay est en quelque sorte l'unique graphe de relations de voisinage du diagramme de Voronoi.

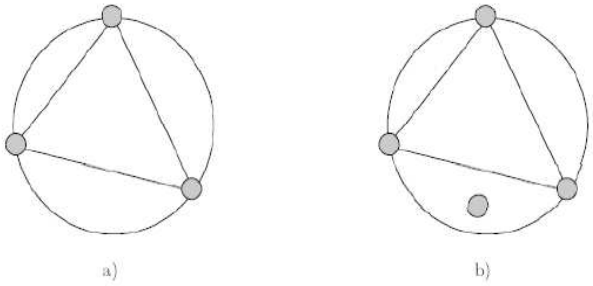


FIGURE 2.12 – Cercle circonscrit aux 3 points - a) Triangulation de Delaunay. b) Un point appartient à la surface décrite par le cercle, ce n'est pas une triangulation.

2.8.1.3 Graphe de Gabriel

Une autre méthode de construction d'un graphe de voisinage est celle du graphe de Gabriel. Elle est proposée, pour la première fois par [Gabriel 1969]. Le graphe de Gabriel est construit comme suit : deux données d_i et d_j sont voisins si l'hypersphère de diamètre | d_i - d_j| qui passe par i et j est vide. Ce type de graphe de voisinage intervient dans un contexte de fouille géographique (mesure de variations) [Hacid 2005] dont la construction doit respecter la définition suivante :

Définition 4. Considérons un graphe G(V, E). Soient deux points v_i et v_j de V. v_i et v_j sont reliés par une arête si seulement si l'hypersphère H(v_i , v_j) de diamètre (v_i , v_j) ne contient aucun autre point de V .

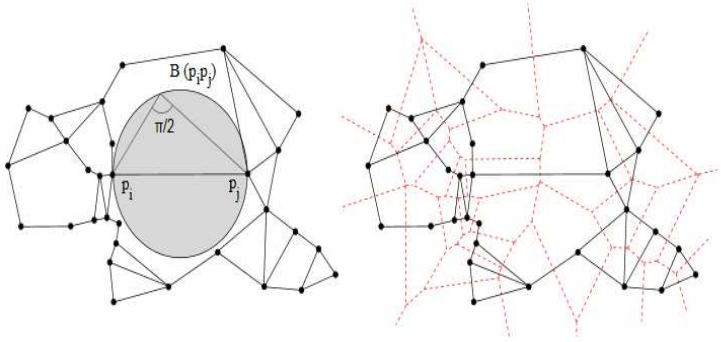


FIGURE 2.13 – Graphe de Gabriel de l'ensemble de points P.

2.8.1.4 Graphe de voisins relatifs

Le Graphe de Voisinage Relatif (RNG) est défini par la propriété suivante : deux données d_i et d_j sont des voisins si l'intersection entre les deux hypersphères respectivement centrées sur d_i et d_j , et avec un rayon de | d_i - d_j |, est vide. Un nouveau principe de construction de graphe de voisinage est proposé par Toussaint au début des années 1980 [Toussaint 1980]. Ce modèle, issu de la géométrie computationnelle, introduit un nouveau modèle de calcul du voisinage tenant compte de la proximité entre les points. Cette notion de proximité dépend de la distance euclidienne entre les points. Le graphe des Voisins Relatifs (VR) s'obtient si la définition 5 est respectée.

Définition 5. Considérons un graphe G (V, E) et deux sommets v_i et v_j de V . La condition suivante définit la construction d'une relation de voisinage selon VR entre ces deux points : v_i et v_j sont reliés par une arête (i.e. voisins) si l'intersection des hypersphères H (v_i, v_j) et H (v_j, v_i) centrées respectivement en v_i et v_j et de rayon |v_i - v_j | est ∅. Les graphes de voisinages, bien

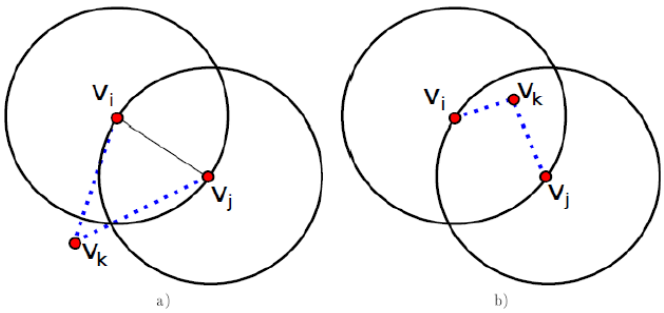


FIGURE 2.14 – Définition des Voisins Relatifs - a) L'intersection des hypershères H(v_i, v_j) et H(v_j, v_i) est vide. Les sommets v_i et v_j sont voisins. Cette relation est concrétisée, sur la figure, par l'existence d'un lien en trait plein entre v_i et v_j . b) Le sommet v_k ∈ à l'intersection des hypersphères de centre v_i et v_j et de rayon |v_i - v_j|. Les points v_i et v_j ne sont donc pas voisins.

que très utiles et adaptés à la résolution d'un grand nombre de problèmes, possèdent quelques inconvénients majeurs. Un des inconvénients concerne directement la complexité induite par la propriété de construction. Sachant que l'algorithme de construction des graphes de proximité a une complexité Θ(n3). Pour un petit ensemble de points à considérer, le coût est relativement faible et négligeable. Mais quand on commence à devoir traiter un ensemble de plusieurs milliers points, la construction coute « cher » en termes de temps de calcul et d'espace mémoire occupé. Nous pouvons citer à titre dexemple

les contraintes les plus couramment rencontrées dans le domaine :

– Importante complexité du problème,
– Temps de calcul de la visualisation,
– Temps d'affichage (successions et grand nombre d'opérations de dessin, 2D et 3D),
– Quantité de mémoire utilisée (mémoire vive et disque),
– Difficulté à maintenir le respect des contraintes esthétiques (l'information perçue dans les graphes devient moins explicite).

En fouille de données traditionnelles, les graphes de voisinage servent à répondre à de nombreuses problématiques telles que la classification et l'interrogation par le contenu d'une base de données multimédia d'images [Hacid 2006]. Les auteurs proposent l'utilisation d'un graphe de voisinage pour l'exploration d'une base de données où les individus sont des images. En effet, l'utilisation d'un graphe de voisinage pour une base de données permet de proposer une structure d'indexation efficace des images (en fonction de leur caractéristiques) pour la recherche d'information. L'intérêt du graphe de voisinage est double. Il doit permettre, dans un cadre de classification supervisée, d'ajouter une nouvelle image en fonction de son voisinage potentiel dans le graphe. Ainsi, cette image est étiquetée selon ce voisinage partageant des caractéristiques similaires voire identiques. L'autre intérêt concerne l'interrogation de la base d'images. L'utilisateur ne fournit pas un vecteur de caractéristiques qu'il lui faudrait déterminer, mais il propose simplement en entrée une image requête pour laquelle le graphe de voisinage doit être capable de fournir une ou plusieurs images similaires (i.e. une image et son voisinage direct). En effet, rechercher la similarité de l'image requête avec les individus (images) du graphe peut être vu comme déterminer son voisinage au sein du graphe. Ainsi, l'utilisateur obtient par recherche dans le graphe de voisinage une ou plusieurs images semblables à son image requête.

Il existe de nombreux algorithmes de visualisation de graphes. Un premier receuil [Eades 1989], sous la direction de Eades, recense près de 180 publications, aussi bien théoriques qu'applicatifs, dans le domaine du dessin de graphes. Un second ouvrage, co-dirigés par Battista et Eades, enrichit le travail précédent et contient une nouvelle bibliographie annotée [Di Battista 1993a], [Di Battista 1994] sur les méthodes de dessin de graphes. Plus récemment, une nouvelle synthèse [Di Battista 1997] met en valeur les points communs des nombreuses recherches effectuées dans le domaine. Nous pouvons citer à titre dexemples : le problème récurent du croisement des arêtes dans le dessin des graphes planaires hiérarchiques ou non, l'interet grandissant de la 3D

afin de répondre à de nombreux problèmes rencontrés en 2D, un référencement des premiers systèmes, toolkits et frameworks pour la production de visualisation de graphes(3DCube [Patrignani 1997], ArchE [Hundack 1997], GRID [Didimo 1997] et Oline Animated Graph Drawing [Eades 1997]). Un ouvrage de référence [Di Battista 1993b] pour la communauté est finalement édité et présenté une revue plutôt exhaustive des différentes algorithmes de dessin de graphes arguant de l'importance du critère esthétique.

Un tel critère valorise une compréhension immédiate du graphe et des structures sous-jacentes qui existent en son sein. Un second ouvrage [Kaufmann 2001], plus récent, est une introduction au vaste monde de graphes (théorie des graphes, algorithmes de graphes, algorithmes de dessin de graphes, aspects topologiques en rapport avec la perception et la langage visuels, la visualisation d'information et les interactions homme- machine. Des graphes dont la structure est simple possèdent souvent une structure arborescente en leur sein. Certaines heuristiques consistent à extraire un arbre de recouvrement du graphe que l'on cherche à visualiser et à réaliser une première disposition des noeuds. Une vue radiale est l'une des méthodes de placement usitées pour cette disposition initiale des nuds. Les arêtes manquantes du graphe par rapport à l'arbre sont ensuite ajoutées. Dans ce cas, le choix d'une bonne disposition des noeuds sur chaque cercle concentrique de la vue radiale doit limiter les croisements d'arêtes afin d'améliorer la lisibilité du graphe [Yee 2001]. Des optimisations existent en ce sens et tentent de proposer une plus faible complexité à la résolution de ce problème [Martí 2003]. Cependant, cette méthode ne convient pas non plus pour la visualisation de grands graphes.

Nous allons nous intéresser aux grands graphes de voisinage qui sont utilisés dans de nombreux domaines. Puisque les méthodes standard pour la construction d'un tel graphe ont une grande complexité. La classification de données à base de graphes est un domaine riche en méthodes et algorithmes. Actuellement, il existe très peu de méthodes qui prennent en compte nativement le problème de la classification incrémentale d'un grand volume de données à l'aide d'un graphe. Ainsi dans [Saha 2006], les auteurs proposent un nouvel algorithme de classification à base de graphes dont la conception a été orientée pour gérer pleinement le cas incrémental. Il travaille sur des graphes non orientés. Cet algorithme peut maintenir de manière efficace les clusters tout en assurant une gestion optimale des ajouts et suppressions de noeuds et d'arêtes. Il gère ainsi les graphes dynamiques. Les graphes de voisinage sont des outils pour représenter et structurer l'information relationnelle au sein d'un espace [Zighed 1999]. Pour obtenir plus d'information sur les domaines d'application de la visualisation de graphe dans le cadre de

l'analyse du web, on peut se référer à Munzner [Munzner 1998], pour son usage en bio-informatique on se référera à Robinson [Robinson] ou à Guelzim [Guelzim 2002].

Les graphes de voisinages interviennent dans de nombreux domaines du fait de leurs qualités intrinsèques. Cela concerne aussi bien la fouille de données traditionnelle [Zighed 2002], la fouille de données spatiales [Aufaure 2000], que la reconnaissance des formes [Ango. 1997]. Leurs propriétés générales leur confèrent une polyvalence et une efficacité pour servie, entre à l'élaboration doutils dexploration (multimédia [Hacid 2006], géographique [Tran Dac 2004]), ou simplement pour de la comparaison entre images [Jolion 2003]. Dans [Ester 1997], les auteurs ont intégré les avantages des graphes de voisinage pour exploiter au mieux les relations de voisinage au sein d'un système de gestion de base de données géographiques. Un autre exemple, en réponse à un problème de classification, porte sur la construction d'une mesure de séparabilité des classes à l'aide de graphes de voisinage [Zighed 2006], [Zighed 2005].

Lavergne et al. [Lavergne 2008a] proposent une méthode de construction incrémentale de graphes de voisinage par des fourmis artificielles. Ils s'inspirent du comportement d'auto-assemblage observé chez des fourmis réelles pour construire de manière incrémentale un graphe de voisinage avec des fourmis artificielles à partir d'une mesure de similarité entre les données. Un graphe construit peut visualiser la forme globale d'un graphe de données et explorer localement les relations de voisinage avec rendu de la proximité réelle entre les données. Il bénéficie ainsi d'une navigation guidée par le contenu. Ils proposent également une extension de cette méthode pour la construction et l'exploration interactive de grands graphes de voisinage.

2.8.2 CHAMELEON

L'algorithme CHAMELEON [Kerr 2000] est une méthode Ascendante Hiérarchique basée sur une représentation sous forme de graphe des k plus proches voisins (voir la figure 2.15). La particularité de l'algorithme est de modéliser à la fois l'inter-connectivité relative RI et la proximité relative RC entre deux sous-graphes pour décider de leur fusion au cours du processus de clustering. RI est globalement une mesure de la force totale des connexions entre les sous-graphes alors que RC est une mesure de la force moyenne de ces connexions. Contrairement à la plupart des autres algorithmes hiérarchiques, CHAMELEON est capable de s'adapter à différentes formes et densités dans les clusters, ou à la présence de bruit, ce qui en fait une méthode

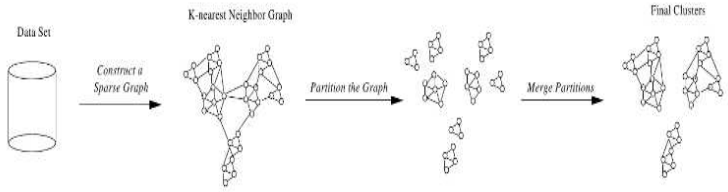


FIGURE 2.15 – Processus général d'une exécution de l'algorithme CHAMELEON.

potentiellement très robuste pour l'analyse de données réelles. Il n'est donc pas adapté à l'analyse de bases de données de grande taille. Ses performances sur des bases de données réelles de grandes dimensions sont inconnues.

2.8.3 Propagation d'Affinité

L'algorithme de Propagation d'Affinité est un autre algorithme très populaire se basant sur une représentation sous forme de graphe [Frey]. L'idée principale est de partir d'un graphe (souvent complet) des données et de faire transmettre des messages entre les données le long des arrêtes en fonction de la valeur de ces arrêtes c'est à dire la similarité entre les données, ici en valeurs négatives (voir la figure 2.16) . Le principal avantage de la Propa-

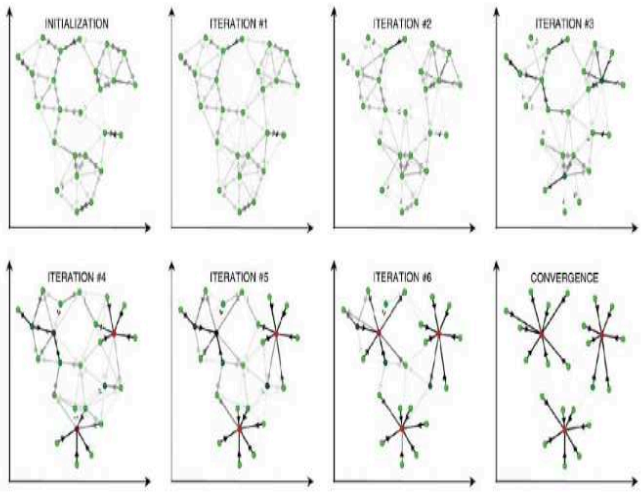


FIGURE 2.16 – Exemple d'une exécution de l'algorithme Propagation d'Affinité [Frey]. Plus un point est rouge, plus il est candidat pour être un représentant. La couleur des arêtes est proportionnelle à la force des messages transmis.

gation d'Affinité est qu'il n'est pas nécessaire de définir à priori le nombre de clusters que l'on souhaite obtenir. Cependant le principal inconvénient de

cet algorithme est sa lenteur. En effet une telle complexité limite l'analyse à des ensembles de données de petites tailles et n'est pas adaptée aux grandes bases de données.

2.8.4 NG + CHL : Méthode à deux niveaux basée sur les graphes

Certaines méthodes à deux niveaux se basent sur la création d'un graphe à partir d'un ensemble de prototypes représentatifs des données.
Le premier niveau (représentation et compression des données) est assuré par l'algorithme NG [Martinetz 1991] qui calcule un nombre important de prototypes à partir des données. Le deuxième niveau se fait pendant l'apprentissage des prototypes, cette étape est effectuée par l'algorithme CHL [Martinetz 1993] ce qui minimise la perte d'information lors de leur classification. L'idée principale est simple : il s'agit de créer des connexions entre les prototypes qui représentent le même type de données et/ou de détruire les connexions reliant des prototypes représentant des données non similaires (voir la figure 2.17). Cette méthode a de nombreux avantages. En particulier

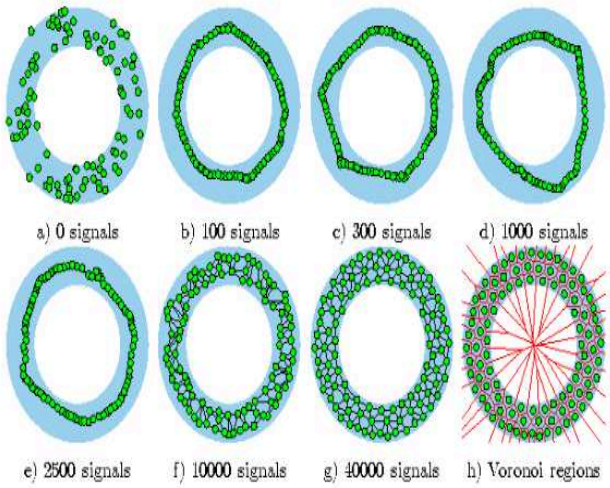


FIGURE 2.17 – Quelques étapes de l'apprentissage par Neural Gas + CHL. Les données sont dans la zone bleue, les prototypes sont en vert. Les connexions pertinentes sont créées au fur et à mesure. A la fin de l'apprentissage les régions de Voronoi déterminent quel prototype sera le plus représentatif de chaque donnée.

elle découvre les clusters en cours d'apprentissage et à partir des données, sont à priori sur leur répartition ou leur structure (comme par exemple le nombre de clusters attendu). Par contre les résultats sont très sensibles à la

présence de bruit dans les données, qui a tendance à favoriser la création de connexions non pertinentes. Enfin, on n'a aucune information sur la structure générale des clusters entre eux, on ne sait pas par exemple si le cluster 1 est plus similaire au cluster 2 que le cluster 3.

2.8.5 Méthodes biomimétiques et les graphes

Nous proposons une présentation des méthodes biomimétiques en utilisant la notion des graphes. Nous pouvons citer à titre d'exemple les travaux de [Garzon 1993] sur la modélisation artificielle d'une molécule, de la conception d'un formalisme pour la biologie moléculaire avec [Danos 2005], [Tarissan 2006]. Les observations faites permettent de produire des modèles artificiels et informatiques pour la résolution de problèmes complexes (i.e. auto-assemblage de molécules in vitro avec [Garzon 1999] et [Beech 2004] ou encore permettre la construction de structures moléculaires à une échelle nanoscopique [Angelov 2006] en travaillant avec de l'ADN [Winfree 1998]). Cependant, d'autres domaines bénéficient également de l'utilisation des graphes tels que la robotique distribuée [Klavins 2002], [Klavins 2004]. De plus amples informations peuvent être trouvées dans [Krasnogor 2005].

Des travaux [Di Caro 1997], [Di Caro 998a], [Di Caro 998b] présentent l'utilisation d'agents-fourmis pour le routage de données dans un réseau. Ce réseau peut être modélisé par un graphe de N noeuds dont les arcs sont pondérés par une largeur de bande. Chaque noeud possède un tampon d'entrée. Quand un paquet arrive sur un noeud, il est dirigé sur un noeud voisin en fonction de son noeud de destination et de la table de routage du noeud sur lequel il se trouve. Le même type de problème est abordé dans [Bonabeau 1998], [Heusse 1998]. Un problème similaire de routage entre des satellites de télécommunication a été proposé [Sigel 2000]. Les fourmis n'accompagnent cependant pas chaque paquet mais sont lancées régulièrement à travers le réseau. La fréquence de mise à jour des tables de routage ne dépend pas dans ce cas du traffic réel. Enfin, des travaux mettant en oeuvre des algorithmes à base de fourmis artificielles ont aussi été menés sur le problème du routage dans les réseaux commutes [Schoonderwoerd 1997]. Tarissan et al. [Tarissan 2006] décrit un réseau d'agents peut être vu de manière abstraite comme un réseau de graphes connectés entre eux.

Dans [Lavergne 2006], l'auteur décrit les différentes approches de l'auto-assemblage rencontrées chez les fourmis et leur capacité à construire des structures complexes à partir d'une population d'agents et d'un ensemble de règles simples. Il s'intéresse à l'auto-assemblage de graphes dans le vivant et

à ses modélisations artificielles. Nous avons synthétisé l'idée que les graphes pouvaient être des structures complexes composées par connexion de graphes intermédiaires. Ces mêmes graphes intermédiaires étant eux-mêmes constitués par auto-assemblage d'entités élémentaires. Nous présentons quelques exemples d'application concrètes d'utilisation du phénomène d'auto-assemblage avec des graphes [Krasnogor 2005].

2.8.5.1 Classification incrémentale avec des graphes

Nous présentons les algorithmes les plus répandus utilisant la structure d'un graphe. Les approches spectrales représentent l'une des familles les plus représentatives de ce type de classification [Jouve], [Kannan 2000], [Kuntz 2000]. Le principal avantage de l'analyse spectrale est qu'il est possible de détecter des clusters de formes arbitraires. En effet de tels clusters sont représentés dans le graphe comme des composantes fortement connectées. En plus, la complexité des calculs mis en jeux ne permet pas d'utiliser une analyse spectrale pour de grandes bases de données. D'autres approches, quant à elle, utilisent largement les chaînes de Markov [Brandes 2003] qui permettent le plus souvent une optimisation des temps de traitement. Nous pouvons citer les travaux de [Dhillon 2005] qui proposent une procédure rapide de classification en se basant sur une modélisation d'un partitionnement multi-niveaux popularisé par l'approche METIS [Karypis 998a]. L'idée est de considérer une transformation d'un graphe initial (correspondant à un partitionnement) en un graphe de taille réduite, que l'on étiquette, et d'étendre ensuite par retour cet étiquetage au graphe initial. ces approches conservent l'inconvénient majeur de devoir considérer la totalité des noeuds/arêtes du graphe.

Lavergne et al. [Lavergne 2007] ont ainsi mis en évidence l'intérêt de l'auto-assemblage dans la construction de structures complexes telles que des graphes avec une dynamique incrémentale (les graphes pouvant toujours évoluer). Un des problèmes souvent considéré est celui de la classification du Web en communautés [Flake 2000]. Il se représente généralement sous la forme d'un graphe où un noeud est une page web et une arête de ce graphe représente un lien entre deux pages. Un autre exemple est celui de la base des citations de CiteSeer [Cit]. Des publications sont mises en ligne sur le web. Des liens entre ces articles sont établis en utilisant citations, co-citations ou similarités entre les textes. Un graphe est ainsi construit où les articles sont les noeuds et les liens représentent les citations d'une publication dans une autre. La classification incrémentale peut également être utilisée pour améliorer le routage de paquets au sein de réseaux ad-hoc [Younis 2004]. L'allocation de ressources dans les réseaux mobiles est un autre exemple où la

classification incrémentale avec des graphes est très utile. Son utilité permet de résoudre des problèmes de bande passante en réponse à des modifications topologiques de la structure du réseau (i.e. mouvement, dysfonctionnement, attribution de noeuds) [Lin 1997].

Ainsi dans [Saha 2006], les auteurs proposent un nouvel algorithme de classification à base de graphes dont la conception a été orientée pour gérer pleinement le cas incrémental. Dans [Flake 2003], l'auteur propose un algorithme qui réalise un partitionnement en basant sur la résolution d'un problème de coupe minimum et réalise de concert une classification dont la qualité satisfait la mesure bi-critère issue de ces mêmes travaux. Cet algorithme a été testé sur de grands graphes avec de bonnes performances en terme de temps d'exécution et de qualité de classification en comparaison avec [Dhillon 2005]. Cependant, la plupart de ces approches considèrent les centres de classes et ne sont pas adaptées à une visualisation des relations de voisinage entre les données. D'autre part, ces méthodes connaissent des complexités relativement élevées et considèrent généralement un grand nombre de noeuds. A cet effet, nous proposons dans nos travaux de thèse une méthode biomimétique de partitionnement d'un volume de données adaptée à une visualisation des relations de voisinage entre les clusters des données.

2.9 Conclusion

Ce chapitre représente un panorama synthétique des différents algorithmes de classification non supervisée. Il y a plusieurs algorithmes de classification des données. Ils diffèrent par la nature de données qu'ils traitent (données numériques ou données de catégorie, petit jeu de données ou gros jeu de données, données de dimension élevée ou moins élevée, sur un flux de données, ...), par les méthodes de distribution des données en classes, par la représentation des classes. Nous avons présenté dans ce chapitre les principes des algorithmes traditionnels pour le problème de classification. Comme la plupart des activités scientifiques, l'essor des différentes techniques de classification a largement bénéficié de l'avènement et du perfectionnement des outils informatiques.

De nos jours, la classification est une démarche qui est appliquée dans d'innombrables domaines. Un autre nom possible pour cette branche de la recherche est la typologie, et la science qui lui est associée est la taxonomie. Mais elle propose aussi de nouvelles sources d'inspiration, car le problème de la classification se rencontre souvent chez les animaux et dans les sys-

tèmes biologiques, une des sources d'inspiration utilisées dans la branche de l'intelligence artificielle est la modélisation de systèmes complexes naturelles à partir de l'intelligence collective.

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 et cela pour plusieurs raisons. D'une part, ce sont des systèmes auto-organisés et non-centralisés. Nous venons de réaliser un tour d'horizon des principales méthodes de classification à base de fourmis artificielles qui regroupent de nombreux avantages des approches biomimétiques. L'un de ses avantages concerne la visualisation de la classification qui dans notre cas s'avère essentiel puisque nous désirons visualiser l'ensemble des données sous forme de graphe de voisinage. Du fait, nous proposons dans les chapitres suivants, deux nouvelles heuristiques, appelées CL-Ant et CL-AntInc, qui vont tenter de répondre au problème de classification non supervisée non hiérarchiques en se basant sur le système de reconnaissance chimique des fourmis et l'utilisation de phéromones.

CHAPITRE 3

L'ALGORITHME INITIAL CL-ANT

Sommaire

3.1 Introduction 90
3.2 Le principe général de l'algorithme CL-Ant 91
3.2.1 Fourmis artificielles : principes 93
3.2.2 Description de l'algorithme 95
3.3 Expérimentations et résultats 104
3.3.1 Mesures d'évaluation 104
3.3.2 Les jeux de données et la mesure de similarité 106
3.3.3 Résultats 110
3.4 Visualisation des résultats 113
3.5 Conclusion 116

Résumé

Nous présentons un nouvel algorithme de classification non supervisée non hiérarchique de construction de graphes dynamiques appelé CL-Ant. Il utilise le principe du système de reconnaissance chimique observé chez des fourmis réelles qui est présenté dans le chapitre précédent appliqué sur des graphes de clusters.
Nous détaillons l'algorithme et les règles locales du comportement des fourmis artificielles et l'évolution qui en résulte sur le graphe de clusters (section 3.2) à partir de l'étude qui a été réalisée dans le chapitre 2 sur les mécanismes fondamentaux du système de reconnaissance chimique chez les fourmis réelles.
Nous décrivons l'ensemble des tests qui ont servi à valider l'algorithme CL-Ant, en étudiant plus particulièrement les résultats obtenus sur des jeux de données numériques et binaires réelles, ainsi que les indices d'évaluation afin de juger la qualité de la classification obtenue (section 3.3).
Dans la partie visualisation des graphes générés, nous allons présenter les graphes de voisinage de quelques bases de données testées par l'algorithme CL-Ant (section 3.4).
Nous concluons en présentant les intérêts de cette approche ainsi les extensions possibles qu'on peut proposer (section 3.5).

3.1 Introduction

Nous proposons dans ce chapitre une nouvelle modélisation qui a été appliquée à la résolution d'un problème majeur en informatique. Ce nouveau modèle se place dans la lignée des travaux précédents sur les algorithmes de classification s'inspirant du comportement des fourmis réelles et plus généralement des systèmes biologiques (voir chapitre 1). Il s'agit de modéliser la manière dont les fourmis forment des groupes possèdant la même odeur coloniale (voir chapitre 2) et d'utiliser ce comportement pour organiser les données selon un graphe qui se construit de manière évolutive. Ces algorithmes peuvent bénéficier de propriétés intéressantes comme celles de produire une optimisation globale de la classification évitant les minimum locaux (construction probabiliste) et dans un temps de calcul «raisonnable», de traiter les données réelles ou encore la nécessité d'informations préalables (nombre de classes, classification initiale) à partir d'une initialisation par l'algorithme K-Means [MacQueen 1967].

Le principe utilisé est le suivant : intuitivement, chaque fourmi/donnée est située au départ dans un cluster, une fourmi extraite de la liste des fourmis appartenant à un nid/cluster, décide de quitter son groupe et de s'intègrer à un autre nid/cluster voisin qui est le plus similaire. Le comportement des fourmis consiste à se déplacer d'un cluster à un autre. Ce comportement est déterminé notamment par la similarité entre les données et le seuil d'acceptation du nid/cluster. Il en résulte une structure de graphe dynamique des données dont les propriétés vont nous permettre de traiter de nombreuses applications : détermination automatique d'un partitionnement (que nous utilisons notamment à titre de comparaison avec d'autres méthodes), et enfin un aperçu visuel de graphe.

Ce chapitre se rapporte donc à la présentation de la première version de l'algorithme de classification non hiérarchique et non supervisée nommé CL-Ant. Nous détaillerons l'algorithme pour former une structure de graphe et les règles locales de comportement des fourmis artificielles à partir de l'étude qui a été réalisée sur le comportement de construction d'une odeur coloniale chez les fourmis réelles. Ensuite, nous présenterons l'ensemble des tests qui ont servi à valider l'algorithme CL-Ant, en étudiant plus particulièrement les résultats obtenus sur des jeux de données numériques et catégorielles/binaires réelles, ainsi que les indices d'évaluation sur la qualité de la classification obtenue. Nous étudierons dans les prochains chapitres l'extension de cet algorithme qui traite les flux de données d'une manière incrémentale. Nous comparerons ensuite les algorithmes proposés à l'algorithme K-Means, à la Classification Ascendante Hiérarchique (CAH) et aux différents algorithmes incrémentaux comme D-Stream, DenStream et ClusStream.

3.2 Le principe général de l'algorithme CL-Ant [1]

Les fourmis vivent regroupées au sein de colonies, elles représentent un super-organisme, dont elles se distinguent des autres individus qui n'appartiennent pas à la colonie. Ce mécanisme a pour rôle principal la défense, et par conséquent le développement et le suivie de la société contre les intrusions provenant de l'extérieur : autres colonies de fourmis, prédatrices, parasites.
A notre connaissance, les modèles sur les quels nous pouvons baser nos tra-

1. Nesrine Masmoudi, Hanane Azzag, Mustapha Lebbah, Cyrille Bertelle : Clustering using chemical and colonial odors of real ants. Proceedings of the 5th World Congress on Nature and Biologically Inspired Computing NaBIC 2013 : 207-213, USA-Fargo, (2013).

vaux sont ceux de Carlin et Hölldobler [**?**, Carlin 1987, **?**] et ceux du Nicolas Labroche [Labroche 2003]. Pour le premier modèle décrit les discriminateurs intrinsèques et extrinsèques participant au mécanisme de reconnaissance des individus rencontrés. Il a été élaboré en prenant en considération les données collectées concernant les fourmis ainsi que d'autres insectes sociaux. Dans ce modèle, une fourmi est principalement représentée par un label cuticulaire («label») qui est l'odeur chimique qu'elle émet, ainsi qu'un modèle de reconnaissance ou («Template») qui correspond selon les auteurs, à un ensemble de discriminateurs appris par la fourmi et représentatifs de sa colonie. Carlin et Hölldobler ajoutent dans [**?**, **?**] que les critères principaux de la discrimination sont :

– L'influence de la reine, dans les petites et grandes colonies (à condition que son odeur soit suffisamment forte et établie),
– L'information génétique propres à chaque fourmi,
– Les odeurs environnementales qui regroupent à la fois l'alimentation et les matériaux utilisés pour la construction du nid.

Ce modèle explique le principe de la reconnaissance chimique entre les fourmis, mais il reste insuffisant pour faire une modélisation dans un similateur de vie artificielle car il n'y a aucune formalisation des relations décrites et certains concepts comme la création de la colonie, l'influence de la reine, de l'alimentation ou l'environnement sur les odeurs des ouvrières ne sont pas explicites. Compte tenu de ces informations manquantes et sur ces limites, Nicolas Labroche propose son propore modèle [Labroche 2003] qui simule la reconnaissance chimique comme une intervension d'une part un ensemble d'organes et d'autre part un ensemble de règles comportementales. Ces dernières président à l'élaboration des odeurs coloniales en autorisant, selon le degré de similitude que deux fourmis observent en se rencontrant, des échanges plus au moins importants de substances chimiques. Ceux-ci ont pour conséquence un rapprochement des odeurs des deux insectes. Cette notion de similitude sera exprimée, Dans cette approche, une mesure de similarité et un ensemble de seuils pourront être utilisés pour quantifier le rapprochement à effectuer entre les individus (voir le chapitre 2).

Nous présentons un nouveau modèle appelé CL-Ant inspiré du comportement des fourmis réelles et plus précisément l'utilisation d'une odeur coloniale comme un code chimique pour classer les fourmis appartenant à différentes colonies. L'algorithme est une modélisation de la façon dont les fourmis reconnaissent l'odeur de leur nid qui leur permet de se protéger elles-mêmes, en reconnaissant et en rejetant les intrus. Notre modèle repose sur le

partage d'une odeur commune coloniale à toutes les fourmis du même nid et l'usage d'une substance volatile appelée phéromone que les fourmis réelles utilisent comme une trace pour marquer leur chemin vers leur nid.

3.2.1 Fourmis artificielles : principes

Pour obtenir une classification de données, nous allons construire un graphe dont les noeuds sont connus a priori à partir d'une étape d'initialisation par l'algorithme K-Means et représentent les clusters des données, et dont les arcs sont présentés pour identifier la relation de voisinage entre les groupes de données. Notons que le graphe construit est un graphe complet et dynamique contiendra des données à tous les noeuds. Nous montrerons dans les résultats de la section 3.4.2, la visualisation des graphes après chaque étape de l'algorithme. Chaque noeud représente le groupe de données à classer. Nous considérons que nous disposons d'une mesure de similarité Sim(W_i, W_j) qui prend comme paramètre un couple de barycentres des clusters CL_i et CL_j. Nous ne faisons pas d'autres hypothèses sur la représentation de données qui peuvent être de tous types de représentation (numériques, catégorielles), il suffit que l'on puisse définir une mesure de similarité.

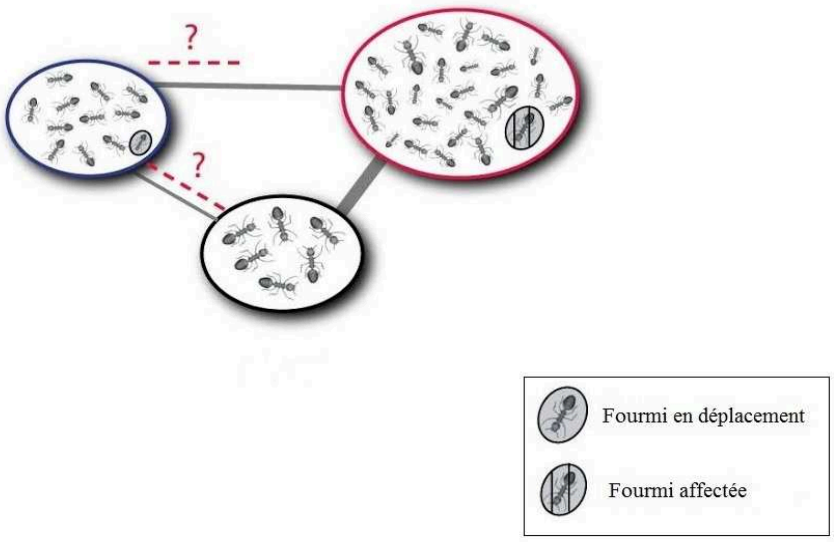


FIGURE 3.1 – Construction du graphe par des fourmis : principe général. Les fourmis qui sont en déplacement sont représentées en gris et les fourmis affectées sont hachurées. Les autres fourmis ne sont pas encore testées.

Le principe de l'algorithme CL-Ant est le suivant : chaque fourmi f_i, i∈{1,...,N} (N est le nombre de données initiale) se situe dans un noeud du graphe, c'est-

à-dire chaque donnée du noeud à classer. Le graphe représente la structure dans laquelle les fourmis vont se déplacer pour construire des groupes similaires de données. Chaque f_i fourmi est simulée, chacune représente une donnée initialement située dans un cluster, et chaque nid représente un cluster. Partant d'une première partition qui est initialisée par l'algorithme K-Means, on attribue les données à chaque classe K dont le centre est le plus proche en terme de distance euclidienne. Cette partition représente la structure graphe complet non orienté, les fourmis vont se déplacer d'un noeud à un autre noeud voisin, puis successivement aux autres noeuds déjà connectés dans le graphe, et ainsi de suite jusqu'à ce que toutes les fourmis soient affectées aux clusters le plus similaires (voir figure 3.1). Tous ces déplacements et ces affectations dépendent de la valeur retournée par la fonction de similarité Sim(f_i, W_j) entre les données, et le barycentre du cluster. Nous voulons utiliser ces comportements pour organiser les données les plus semblables dans le même groupe techniquement appelé cluster. Pour chaque fourmi f_i nous allons donc définir les notions suivantes :

- K clusters CL_j (avec j = 1,...,K) ou noeuds sont initialisés par l'algorithme de K-Means pour construire la structure du graphe. Ce nombre fixé à priori représente les nids formés par des fourmis f_i qui se déplacent d'un nid à un autre nid voisin essayant de trouver celui qui porte la même odeur chimique ;
- chaque cluster CL_j possède un barycentre W_j représentant le centre de gravité de chaque cluster j ;
- une donnée d_i représentée par chaque fourmi et qui représente la donnée à classer (i-ème donnée de la base) ;
- un seuil d'acceptation Th_{CL_j} chaque cluster CL_j permet de définir si la donnée d_i est suffisamment similaire ou bien suffisamment dissimilaire aux autres données représentées par le cluster ;
- considérons également que le voisinage entre deux clusters est représenté par le taux de phéromone Ph_{ij} (avec i, j les indices des deux clusters voisins CL_i et CL_j).

Lors de la construction de la structure (figures 3.1 et 3.2), chacune des fourmis sera soit :

- **en déplacement** sur le graphe. Dans ce cas, nous considérons que chaque fourmi f_i se déplace vers le cluster CL_j : passer d'un nid à l'autre, les fourmis déposent sur le sol une substance appelée phéromone qui crée un certain type d'une piste chimique. Les fourmis peuvent sentir les phéromones qui jouent le rôle de marqueurs de chemin : au moment de choisir leur chemin, les fourmis ont tendance à choisir la voie la plus forte concen-

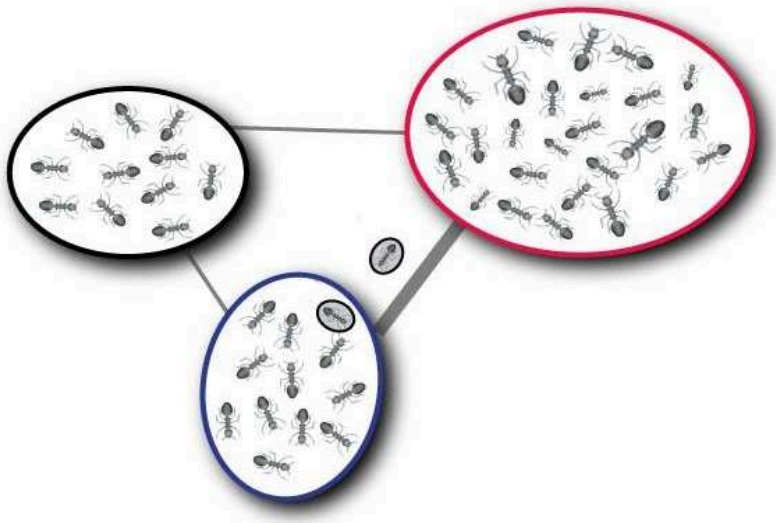


FIGURE 3.2 – Décision d'une fourmi de quitter son cluster d'appartenance. Une fourmi f_i en déplacement est en gris.

tration en phéromones (voir figure 3.2) ;
– **affectée** la fourmi f_i à un nouveau cluster voisin. Une mise à jour des prototypes est effectuée. Le poids de l'arête est calculé à partir d'une distance, selon la nature des données, qui relie les deux centres des deux pôles (source et cible). Ce poids représente le taux de phéromone qui sera augmenter, après l'affectation, entre le cluster source et celui accepté par la nouvelle fourmi comme une nouvelle cible.

3.2.2 Description de l'algorithme

L'algorithme CL-Ant fonctionne de la manière suivante (algorithme 1) : pour un nombre d'itérations fixées, une fourmi f_i extraite de la liste des fourmis va s'intégrer ou bien se déplacer à un autre nid voisin en fonction de la similarité avec son voisinage. Tant qu'il existe une fourmi f_i en déplacement, on simule une action pour f_i en fonction de sa position sur l'arrête du graphe, c'est-à-dire qu'elle a décidé de quitter le cluster source vers un autre plus similaire. Initialement, les fourmis sont placées dans K clusters à partir de l'algorithme K-Means. Elles sont en déplacement dans une structure du graphe dynamique. Le seuil de similarité est calculé selon une distance et qui peut être entre 0 et 1. Pour ces deux paramètres nous citerons dans la section 3.2.2.5, d'autres paramètres mis en place pour l'éxécution de cet algorithme. Les fourmis sont séquentiellement testées. Ensuite pour chaque fourmi f_i, on

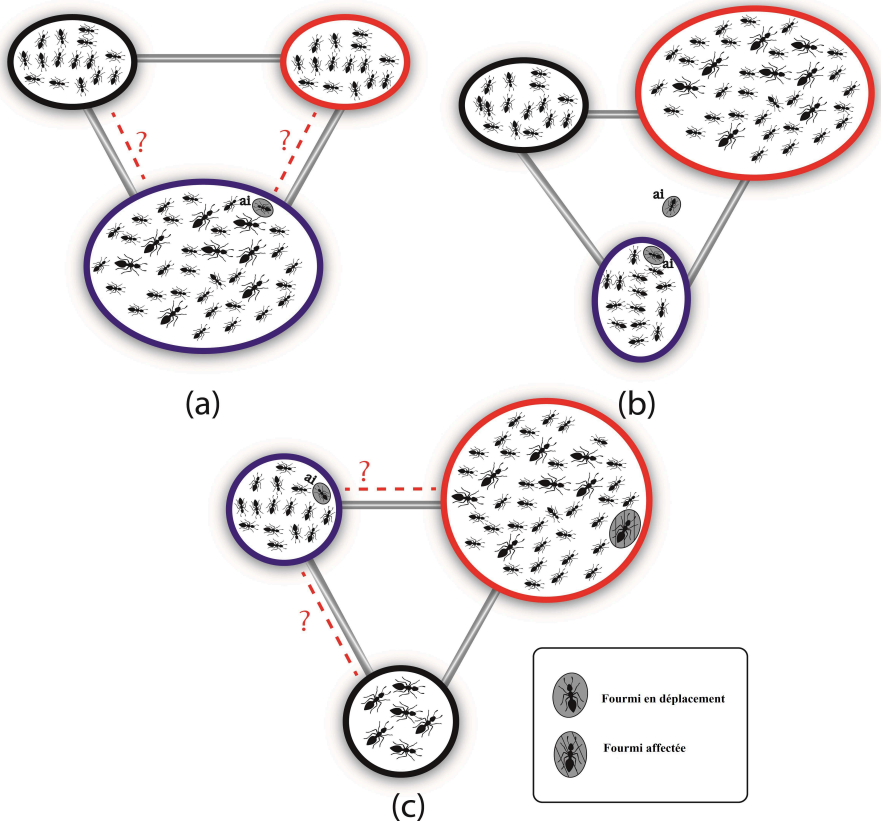


FIGURE 3.3 – Règles de comportement de f_i si elle est affectée au nid/cluster voisin $CL_{j'}$ (voir l'algorithme 2).

distingue 4 étapes :

– **Étape d'initialisation par l'algorithme K-Means :** l'algorithme K-Means [MacQueen 1967] choisit K premiers points représentant les centres de classes K (centre de gravité). Delà, une première partition est formée en attribuant chaque donnée à la classe K dans laquelle le centre est le plus proche en terme d'une mesure de distance (voir figure. 3.3 (a)).

– **Décision de la fourmi f_i de quitter son cluster :** une fois dans son nid (cluster) chaque fourmi f_i a une similarité avec le centre du cluster, si Sim(f_i, W_j)$<$ seuil du nid (cluster) CL_j, où W_j est le centre de CL_j. La fourmi décide de quitter son cluster et chercher un autre cluster qui semble le plus proche (voir figure. 3.3 (b)).

– **Déplacement de la fourmi f_i :** la fourmi f_i quitte son groupe, elle se déplace vers un autre cluster en choisissant la voie qui possède une plus forte

concentration en phéromone. Si elle n'est pas acceptée par un cluster dans son proximité, elle se déplace vers un autre en prenant toujours le chemin le plus dense.

– **Affectation de la fourmi f_i dans le nouveau cluster voisin :** la fourmi est acceptée par le cluster si Sim(a_i, $W_{j'}$) $\geq$ seuil du nid (cluster) $CL_{j'}$ puis elle met à jour l'identifiant de sa classe, et le seuil du cluster destination ainsi que celui du cluster d'origine. Le chemin qui mène la fourmi de son cluster d'origine vers le cluster de destination sera également mis à jour (voir figure. 3.3 (c)).

Nous avons développé un modèle de règles comportementaux des fourmis artificielles pour le clustering non supervisé en utilisant les principes de l'odeur coloniale et les traces des phéromones chez les fourmis réelles. Nous validons cette approche avec une étude comparative dans le chapitre suivant (voir chapitre 5). Les fourmis passent par deux étapes principales : elles empruntent le chemin avec un taux de phéromone maximum, puis une possibilté de s'intégrer dans un nouveau cluster voisin.

Algorithme 1 Principe général de l'algorithme par des fourmis artificielles.

1: Début
2: **Initialiser** les clusetrs par l'algorithme K-Means
3: **Initialiser** les paramètres de la fourmi f_i et les clusters CL_j
   – Calculer la similarité Sim (f_i, W_k) est la distance entre la fourmi f_i et le barycentre W_k du cluster
   – Calculer le seuil d'acceptation du nid (cluster)
   – Spécifier l'identifiant du fourmi
4: **Pour**(un nombre d'itérations) faire
5: **Si** (Sim(f_i, W_j) $<$ le seuil d'acceptation du nid CL_j alors
6: La décision de a fourmi f_i **de quitter** son cluster CL_j) d'appartenance
7: **Déplacement** de la fourmi
8: **Si** $(Sim(f_i, W_{j'}) >=$ le seuil d'acceptation du nid $CL_{j'})$ alors
9: **Affectation** de fourmi f_i au clutser voisin qui lui semble le plus similaire
10: **Mise à jour** (l'identifiant de la fourmi f_i, Sim(f_i, $W_{j'}$), le seuil d'acceptation du nid (cluster) CL_j et $CL_{j'}$, le taux de phéromone du chemin emprunté par la fourmi f_i)
11: Fin

Algorithme 2 Vue détaillée de l'algorithme CL-Ant.

Initialisation : calculer les clusters en utilisant l'algorithme K-Means
Construction d'un graphe complet reliant tous les clusters initialisés par K-Means
Initialisation des paramètres :
α : valeur incrémental du phéromone
γ : valeur d'évaporation du phéromone
Nombre maximal d'itérations NB = 1000
Chaque centroïde du cluster W_j, et chaque seuil du cluster Th_{CL_j}
Le taux de phéromone pour chaque arête Ph_{ij} = 0
Tantque (NB itérations) **{**
Pour chaque fourmi f_i, i $\in \{1, ..., n\}$ {
CL_j est le cluster dont f_i l'appartient
Si(Sim $(f_i, W_j) < Th_{CL_j}$) **alors** {
  Initialiser la liste tabou avec $l = 1, ...k; l \neq j$ correspondant à tous les clusters pour les explorer
  succès = faux
  Tantque (non succès et la liste tabou n'est pas vide) {
  f_i se déplace vers un autre cluster $CL_{j'}$, j' correspond à $argmax = \{Sim(W'_j, W_l) + Ph_{jl}$; l appartenant à la liste tabou }
    Si(Sim $(a_i, W_{j'}) \geq Th_{CL_{j'}}$) **alors** {
      succès = vrai
      Affecter f_i à $CL_{j'}$
      Mise à jour (W_j, $W_{j'}$, Th_{CL_j}, $Th_{CL_{j'}}$)
      Mise à jour du taux du phéromone $Ph_{jj'} = Ph_{jj'} + \alpha$ }
    Si non {
      Retirer j' de la liste tabou }
    Fin Si
    Évaporer le taux de phéromone sur tous les arêtes du graphe $Ph_{hl} = Ph_{hl}(1 - \gamma)$; $1 \leq h, l \leq k$
  Fin Tantque
Fin Si
Si (non succès) **alors**
  f_i reste dans son cluster initial CL_j
Fin Si
Fin Pour
Fin Tantque

3.2.2.1 Principes de construction d'un graphe avec des fourmis

Un système de notation est tout d'abord défini afin de représenter de manière formelle les graphes. Nous disposons d'un graphe G = (V, E) avec V l'ensemble des n noeuds du graphe et E l'ensemble des arêtes associées. Un noeud représente un cluster et une arête est une liaison qui définit la relation de voisinage entre les clusters voisins. On note par CL-Ant$_1()$ l'étape d'initialisation (de l'itérations [1, 3] de l'algorithme 2) et par CL-Ant$_2()$ conformément à l'itérations [4, 27] de l'algorithme 2.

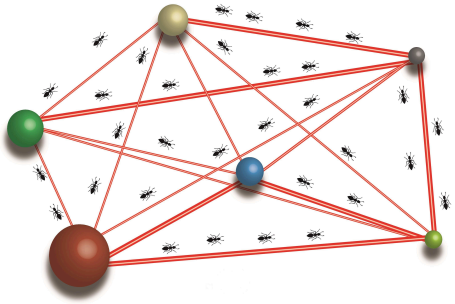


FIGURE 3.4 – Le graphe résultat de l'étape d'initialisation en utilisant l'algorithme K-Means : CL-Ant$_1()$

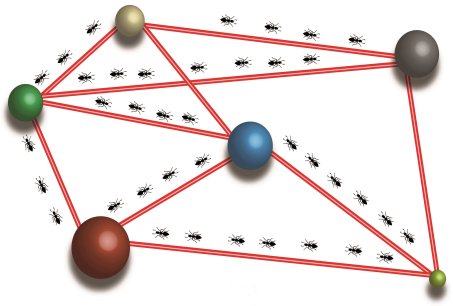


FIGURE 3.5 – La première étape de clustering : CL$-Ant_2()$

Dans un premier temps, la structure est un graphe complet (voir figure 3.4) et après un nombre d'itérations fixe par l'algorithme CL-Ant, le graphe sera réduit (voir figure 3.5). La forte proximité entre les clusters est maintenu tandis que les arêtes qui représentent le taux le plus bas en phéromone seront éliminés. Pour aboutir à cette structure : nous considérons K noeuds comme les clusters à partir de laquelle la structure graphique sera construit. Dans chaque noeud du graphe, nous simulons chaque fourmi f_i : elle se déplace d'un noeud à l'autre jusqu'à ce qu'elle trouve un emplacement idéal pour intégrer. on désigne CL_j par le nid (ou le cluster) dans lequel la fourmi f_i est situé. L'arete relie les deux noeuds est notée $Ph_{jj'}$, et elle présente la relation de voisinage entre les clusters CL_j et $CL_{j'}$. Ainsi, les mouvements des fourmis dans le graphe seront similés suivant le chemin avec des taux de phéromones les plus élevées (à savoir, les voisins qui sont les plus semblables à f_i) : de cette manière, elle atteindra le cluster contenant des fourmis très similaires (voir la section 3.2.2.6).
Pour que la fourmi f_i prend la décision de quitter son cluster ou bien de s'intégrer à un autre cluster de proximité tout est dépend du seuil du nid (cluster) et la similitude entre les fourmis et le centre du cluster. Cette similartité est représentée par une mesure de distance selon le type de données à tester (voir

la section 3.6 pour le calcul de similarité). L'algorithme se termine lorsque toutes les fourmis sont testées pour un nombre d'itérations fixés a priori. Nous considérons maintenant l'aspect algorithmique de la construction d'un graphe de proximité (voir l'algorithme 3), comme mentionné précédemment qui peut être divisé en deux étapes (voir les figures 3.4, 3.5). La première étape de l'algorithme 3 (par exemple, de la ligne 1 à la ligne 5), présente l'étape d'initialisation par l'algorithme K-Means. Pour construire un graphe complet, tous les clusters CL_j doivent être reliés entre eux pour définir la structure de relation de voisinage entre les noeuds du graphe.

Dans la deuxième partie de l'algorithme (par exemple, la ligne 6), lorsque toutes les fourmis sont testées et admises, on obtient un graphe dynamique où chaque noeud est un «cluster» qui peut être affiché avec une plateforme spécifique «Tulip» (voir les détails dans la sections 3.7).

Algorithme 3 Vue générale de l'algorithme non suppervisé de construction non hiérarchique du graphes de voisinage.

1: **Entrée :** D : un ensemble de données où $D = \{d_1, .., d_n\}$
2: **Sortie** : $G_{proximity} \leftarrow$ CL-Ant(D)
3: **Initialisation :**
   – n : le nombre total de données formant l'ensemble D
   – C : clusters après l'algorithme K-Means
   – G, G_c, $G_{proximity}$: graphes résultant après l'exécution de algorithme
   /* CL-Ant$_1$ Selon la règle R1 de l'algorithme CL-Ant */
4: C $\leftarrow$ CL-Ant$_1(D)$
5: G_c graphe complet reliant tous les clusters C
   /* CL-Ant$_2$ selon les règles [2, 6] de l'algorithme CL-Ant */
6: $G_{proximity} \leftarrow$ CL-Ant$_2(G_c)$

3.2.2.2 Étape d'initialisation

Le choix d'utiliser l'algorithme K-Means qui est une technique de partitionnement des plus simples qui puisse être en utilisant l'erreur quadratique comme critère d'évaluation d'une partition. Dans un premier temps, les objets sont regroupés autour de K centres arbitraires : la classe associée au centre est constituée de l'ensemble des points les plus proches de centtre que de tout autre centre. Les centres de gravité sont ensuite calculés à partir des classes qui viennent d'être formées. On recommence l'opération en prenant comme centre de classe les centres de gravité trouvés et ainsi de suite jusqu'à ce que les objets ne changent plus de classe. Comme on a mentionné les

détails dans le chapitre 2.

Le nombre de classes K fixé a priori peut diminuer au cours des itérations : en effet, si une classe n'attire aucun objet, elle sera vide et donc éliminée. Cela peut représenter un inconvénient si l'on désire obtenir K classes non vide. L'algorithme K-Means est une heuristique deterministe nécessite que le nombre de classes soit initialisé, ce qui diminue l'intérêt de la méthode pour un expert cherchant justement à connaître ce nombre de classes. En plus, la complexité de cette méthode est en Θ(KNniter) où niter est le nombre d'itérations effectuées [**?**] et N est le nombre total de données. Le principal inconvénient de l'algorithme K-Means est que la partition finale dépend du choix de la partition initiale. Le minimum global n'est pas obligatoirement atteint, on est seulement certain d'obtenir la meilleure partition à partir de la partition de départ choisie.

3.2.2.3 Propriétés de l'algorithme

Notre algorithme est déterministe. En effet, CL-Ant est un modèle heuristique simple qui peut être dérivé de comportement des fourmis réelles plus particulièrement par les principes des odeurs coloniales et les traces de phéromone. La conception de CL-Ant permet de définir plusieurs propriétés. Tout d'abord, comme cela est représenté sur les figures 3.4 et 3.5, les noeuds sont liés entre eux pour construire un graphe complet G, chaque noeud correspondant à une classe constituée de toutes les données présentes dans un cluster. Cette classification peut être comparée à d'autres obtenues par des algorithmes non hiérarchiques par exemple comme ceux basés sur les centroïdes (voir chapitre 5). Le graphe pourrait aussi être interprété comme un ensemble de clusters de données avec une forte proximité.

De plus, CL-Ant répond aux propriétés visées pour une bonne classification de données représentant une visualisation spécifique par les graphes dynamiques, c'est-à-dire : on commence par un graphe complet pour aboutir enfin à une structure de graphe dynamique où chaque noeud du graphe représente une catégorie composée de toutes les fourmis les plus similaires en terme de distance. Soit CL_j le cluster qui est représenté par un noeud du graphe G. Nous souhaitons que les clusters CL_j et $CL_{j'}$ qui sont liés par une arête épaisse soient les plus similaires possibles entre eux, et les fourmis de f_i qui sont situées dans des clusters CL_j et $CL_{j'}$ soient liés par une arête fine représentent des catégories de données les plus dissimilaires possible entre elles. Autrement dit, la structure de graphe utilisée dans notre approche est constituée de catégories de données homogènes lorsque la relation de voisinage est

forte, et les clusters les plus dissimilaires possible entre elles si la relation de proximité est moins élevée.

Dans la suite de ce document nous allons étudié une autre version qui peut découler de notre modèle afin d'explorer une possibilité de traiter des données en flux. Nous présenterons et nous testerons cette version incrémentale dans le chapitre suivant. Dans la section 3.4, nous avons testé cette première version sur des bases de données numériques et catégorielles (binaires) afin de fixer et valider ses paramètres dans la section suivante.

3.2.2.4 Étude des paramètres de l'algorithme CL-Ant

Notre approche se repose sur le partage d'une odeur coloniale commune à toutes les fourmis du même nid et les substances volatiles, appelées phéromones, qu'elles utilisent comme une trace pour marquer leur chemin vers le nid. Nous proposons d'utiliser ces comportements pour organiser les données les plus semblables dans un même groupe appelé cluster. Les propriétés générales de notre algorithme peuvent également être appliquées à tous les algorithmes de fourmis pour le clustering. Tout d'abord, CL-Ant peut faire face à deux types de données qu'on fournit pour définir une mesure de similarité ou une distance. CL-Ant n'est pas limité à des données numériques par exemple, mais peut être testé à des données catégorielles (binaires) . Nous utilisons la même mesure de similarité Sim (X, Y) entre deux données X et Y, appartenant à un espace métrique :

$$Sim(X,Y) = 1 - \parallel X - Y \parallel^2 \tag{3.1}$$

Les données sont regroupées en clusters, le j^{th} est noté CL_j, localisé par son centre de gravité W_j.
Nous utilisons donc les paramètres suivants :
- f_i, i = 1,...,n sont l'ensemble des fourmis. Chaque fourmi est associé à une donnée ;
- CL_j, j = 1,...,k sont l'ensemble des clusters ;
- W_j est le centre de gravité du cluster CL_j ;
- Ph_{ij} est le taux de phéromone associé à chaque arête ;
- La distance entre W_j et $W_{j'}$ représente la relation de voisinage relative à l'arête e(CL_j, $CL_{j'}$) est calculée dans l'equation 3.1.
- Th_{CL_j} : chaque cluster CL_j possède un seuil d'affiliation défini comme suit :

$$Th_{CL_j} = \frac{\sum_{i=1}^{n}\sum_{j=i+1}^{n}\frac{Sim(x_{f_i},x_{f_j})}{N} + \min(Sim(x_{f_i},x_{f_j}))}{2} \tag{3.2}$$

où f_i et $f_j \in CL_j$, $i \in \{1, ..., n\}$, $N = \frac{n(n-1)}{2}$, x_{f_i} est le vecteur de données associé à la fourmi f_i.

Notre approche a un seul paramètre Ph_{ij}, qui a une influence claire sur le nombre d'arêtes. Lorsque le poids de certaines arêtes devient au dessous d'un seuil, cette arête sera retirée. Le but de notre algorithme est de réduire le graphe initial en éliminant les arêtes avec un taux de phéromone bas et nous gardons la relation de voisinage forte représentée par un taux de phéromone élevé.
Une fourmi initialement en j se déplacera vers le cluster l calculé par :

$$l = argmax_{\{i|i\text{voisin de}j\}}(Sim(W_j, W_i) + Ph_{ji}) \tag{3.3}$$

Le taux de phéromone sur l'arête e(j, l) est mise à jour par la formule suivante :

$$Ph_{jl}(t+1) = Ph_{jl}(t) + \alpha \tag{3.4}$$

où la valeur de α qui se situe dans l'intervalle fermé [0, 1] est le paramètre de renforcement. Après l'étape d'initialisation de l'algorithme K-Means, les arêtes et les noeuds dans le graphe résultant sont définis. Tout en fixant les traces de phéromone, si α = 0, le taux de phéromone est calculé par la mesure de similarité qui concerne le type de données testées entre les deux centres des clusters voisins (noeuds) dans le graphe.

Une fourmi qui se déplace et cherche un nid semblable, c'est-à-dire la fourmi cherche un nid avec sa même odeur coloniale, dépose une substance chimique appelée phéromone le long de son chemin. Comme les voyages de fourmis, il cherche les traces de phéromone pour son chemin préférant. Elles suivent le chemin avec des niveaux plus élevés de dépôts de phéromone. Pendant ce processus, l'évaporation de phéromone est un autre facteur qui influe sur la quantité de dépôt de phéromone, considéré comme un mécanisme d'exploration qui retarde une convergence plus rapide de toutes les fourmis vers la trajectoire optimale. La diminution de l'intensité du phéromone conduit à l'exploration des chemins différents au cours de l'ensemble du processus de recherche. Le mécanisme d'évaporation évite de faire des erreurs ou bien de mauvais choix. Il joue un rôle important à la délimitation de la valeur maximale réalisable par les traces de phéromone. Dans notre approche, l'évaporation du phéromone est entrelacée avec le dépôt de ce dernier par la fourmi. Après le déplacement de chaque fourmi vers un noeud

voisin en fonction de son comportement de recherche, les traces de phéromones sont évaporées en appliquant l'équation suivante pour tous les arêtes :

$$Ph_{hl} = Ph_{hl} \times (1 - \gamma); \qquad 1 \leq h, l \leq k \tag{3.5}$$

où $\gamma \in [0, 1]$ est un paramètre, K est l'ensemble de tous les noeuds du problème. La valeur de l'évaporation du taux de phéromone γ est comprise entre 0 et 1. S'il n'y avait pas d'évaporation du tout, les arêtes choisies par les premières fourmis seraient trop attractives pour les suivantes. Dans ce cas, l'exploration de l'espace de solution serait limitée. Dans la section 3.4.3, nous testerons, sur plusieurs bases de données, différentes valeurs de α et γ qui nous permettons d'obtenir les meilleurs résultats.

3.3 Expérimentations et résultats

Nous présentons dans cette section une étude expérimentale de CL-Ant dans laquelle nous testons différentes valeurs pour le taux de phéromone et le taux d'évaporation afin déterminer l'influence de ces paramètres sur l'algorithme. La première partie se focalise sur la présentation des mesures d'évaluation et des jeux de données utilisés pour évaluer notre méthode. La seconde partie énumère l'ensemble des tests qui ont servi à déterminer la performances de la méthode, et valide les valeurs pour lesquelles l'algorithme obtient la meilleure classification.

3.3.1 Mesures d'évaluation

Nous avons évalué et comparé nos algorithmes sur 19 bases de données, qui ont de N = 148 à 58000 exemples. Table 3.1 résume les caractéristiques de données. Maintenant, la question suivante est de savoir comment mesurer la qualité de classification. Il y a beaucoup de suggestions pour une mesure de qualité [**?**], [**?**]. Une telle mesure peut être utilisée pour calculer la qualité d'un cluster. Plusieurs méthodes prennent en compte une grande similarité dans un cluster et une faible similitude entre les clusters de comparer la qualité de clustering donnée par différents algorithmes de clustering. Notre calcul nécessite la partition des données qui seront comparées.
Nous allons utiliser certains critères bien connus pour mesurer la qualité de clustering. L'évaluation des résultats est effectuée avec un indice de pureté [**?**] et par un autre indice couramment utilisé est l'indice de Rand [**?**]. Nous ajoutons à titre d'exemple l'indice NMI comme un autre critère d'évaluation afin de juger la qualité de classification. L'évaluation de la qualité d'une classification consiste à déterminer si la répartition des objets dans les différents

clusters esr cohérente avec les connaissances disponibles sur les données.
Un critère général pour évaluer les résultats du regroupement, est de comparer la partition calculée avec une partition initiale.
Nous considérons ici que la connaissance est un ensemble d'objets étiquetés. Soit N le nombre dobjets étiquetés, C = $c_1, c_2, ..., c_k$ les clusters trouvés par l'algorithme de clustering, W = $w_1, w_2, ..., w_C$ les classes des objets étiquetés, c_k les objets appartenant au cluster k et w_k l'ensemble des objets de la classe k,$\mid c_k \mid$ le nombre d'objets du cluster k et $n_j^i = \mid W_i \bigcap C_j \mid$ est le nombre d'objets à la fois dans les clusters i et j.

– **L'indice de pureté**
La façon la plus simple de calculer la pureté est de chercher la classe majoritaire dans chaque classe et additionnant le nombre d'objets de cette classe pour chaque cluster. Il prend sa valeur dans [0, 1] ; 1 indique si tous les clusters sont purs (voir l'equation 3.6). On peut définir la pureté comme suit :

$$Purete(C, W) = \frac{1}{N} \sum_{i}^{k} \arg\max_{j}(n_j^i) \tag{3.6}$$

– **L'indice de rand**
Cet indice mesure le pourcentage des paires d'observation appartenant à la même classe, et qui sont attribuées au même groupe. La valeur de Rand varie entre 0 et 1, où 1 signifie que les deux cloisons sont identiques et 0 indique que les partitions n'ont pas des objets communs (voir l'equation 3.7). On dit qu'un couple d'objets est un vrai positif (VP) si les deux objets sont de la même classe et sont placés dans le même cluster, et un vrai négatif (VN) quand les deux objets sont de classes différentes et sont placés dans deux clusters différents. Un faux positif (FP) correspond à deux objets de classes différentes placés dans le même cluster. Un faux négatif (FN) correspond à deux objets de la même classe dans deux clusters différents. On peut définir le rand de la manière suivante :

$$Rand(C, W) = \frac{\mid VP \bigcup VN \mid}{\mid VP \bigcup FP \bigcup FN \bigcup VN \mid} \tag{3.7}$$

$\mid VP \bigcup FP \bigcup FN \bigcup VN \mid$ représentant tous les couples possibles d'objets et $\mid VP \bigcup VN \mid$ les couples d'objets correctement classés. L'indice de Rand donne cependant un poids égal aux faux positifs et aux faux négatifs.

– **L'indice NMI**

Nous présentons l'indice NMI ou Normalized Mutual Information (voir l'equation 3.8) qui peut interpréter théoriquement l'information. Cet indice évalue la qualité de la classification par rapport à un étiquetage de classe des données. Il mesure à quel point l'algorithme de clustering pourrait reconstruire la distribution des données étiquettées [**?**].

$$NMI(C,W) = \frac{I(C,W)}{(H(C)+H(W))/2} \tag{3.8}$$

Où I(C, W) est l'information mutuelle entre C et W (voir l'equation 3.9), H(C) et H(W) les entropies respectives pour C et W (voir l'equation 3.10).

$$I(C,W) = \sum_i \sum_j \frac{n_j^i}{N} \log \frac{n_j^i \times N}{\mid c_i \mid \times \mid w_j \mid} \tag{3.9}$$

$$H(W) = -\sum_k \frac{\mid w_k \mid}{N} \log \frac{\mid w_k \mid}{N} \tag{3.10}$$

3.3.2 Les jeux de données et la mesure de similarité

Nous abordons l'utilisation des bases de données dont les objets étiquetés (on connaît la classe de chaque donnée) pour évaluer la qualité d'un résultat de clustering. Cette classe (classe réelle) n'est bien entendue pas donnée à CL-Ant, nous l'utilisons seulement pour évaluer la qualité de la classification obtenue. Afin d'analyser cet algorithme, nous l'avons évalué sur des données numériques et catégorielles. Nous avons utilisé pour cela 19 bases de données réelles issues du Machine Learning Repository [**?**].
Le tableau 3.1 présente pour chaque base, le nombre de classes réelles (CR), la dimension de l'espace des données (N_{Att}) et le nombre de données total (N). L'algorithme CL-Ant a la particularité de posséder deux paramètres présentés dans le tableau 3.2. Nous avons considéré les valeurs α et γ appartenant à l'intervalle [0, 1] et nous avons choisi trois combinaisons possibles. En effet de faibles valeurs pour α et γ rendraient les fourmis moins exigeantes (très tolérantes) pour le choix du chemin sur lequel elles se déplacent. Si une fourmi n'arrive pas à s'intégrer, elle se déplace vers le chemin le plus similaire. Ce choix est guidé par son seuil de similarité, et par conséquent, une forte valeur pour α et γ risque de déplacer la fourmi vers un chemin comportant des fourmis dissimilaires à elle. La mesure de similarité que nous utilisons pour ces données est une distance euclidienne (voir équation 3.1)

Bases	N	N_{Att}	CR	Type
Iris	150	4	3	numérique
Wine	178	13	3	numérique
Glass	214	9	2	numérique
SpectF Heart	267	44	2	numérique
Haberman	306	4	2	numérique
Blood	748	5	2	numérique
Pima	768	8	2	numérique
SpamBase	4601	57	2	numérique
Semeion	1593	256	2	numérique
WineQuality	4898	12	8	numérique
WaveForm	5000	40	3	numérique
Statlog(Heart)	6435	37	7	numérique
MagicGamma	19020	11	2	numérique
Statlog(Shuttle)	58000	9	7	numérique
Breast Cancer	286	9	2	catégorielle
Spect	267	22	2	catégorielle
Balance Scale	625	4	3	catégorielle
Tic-Tac-Toe Endgame	958	9	2	catégorielle
Lymphography	148	18	4	catégorielle
Primary Tumor	339	17	22	catégorielle

TABLE 3.1 – Jeux de données utilisées dans l'évaluation.

lorsqu'elles sont numériques [2] et une distance de Hamming (voir équation 3.11) lorsqu'elles sont catégorielles [3].

Pour déterminer l'influences de chaque paramètre, nous avons fixé les valeurs de α et de γ suivant :
En statistiques descriptives, il est possible de résumer un ensemble d'observations par des grandeurs caractéristiques. Dans le cas de la distance euclidienne, on peut résumer un ensemble réel par sa moyenne et son écart-type, ou si les observations sont en dimensions multiples par le centre de gravité et son inertie. Des caractéristiques équivalentes ont été définies pour la dis-

2. Nesrine Masmoudi, Hanane Azzag, Mustapha Lebbah, Cyrille Bertelle, Maher Ben Jemaa : How to use ants for data stream clustering. The IEEE Congress on Evolutionary Computation CEC 2015 : 656-663, Japon-Sendai, (2015)

3. Nesrine Masmoudi, Hanane Azzag, Mustapha Lebbah, Cyrille Bertelle, Maher Ben Jemaa : Clustering of Binary Data Sets Using Artificial Ants Algorithm. The 22th International Conference on Neural Information Processing ICONIP (1) 2015 : 716-723, Turkey-Istanbul, (2015)

Tests	α	γ
Test 1	0.01	0.002
Test 2	0.02	0.001
Test 3	0.03	0.001

TABLE 3.2 – Paramètres testés pour CL-Ant.

Individus/modalités	E1	E2	E3
M1	100	1100	11100
M2	110	1111	10000
M3	111	1000	11111
M4	111	1110	11000
M5	111	1111	11000
M6	100	1110	11000
Médiane	111	1110	11000

TABLE 3.3 – Tableau des variables catégorielles codées en additif ; on remarque que dans les trois cas la médiane représente l'une des modalités de la variable.

tance de Hamming, elles permettent de caractériser un ensemble de vecteurs binaires (en dimensions simples ou multiples) à l'aide d'une valeur centrale, elle-même binaire et d'un écart-type. Nous rappelons dans ce qui suit, pour la distance de Hamming, les définitions et les propriétés attachées à ces grandeurs. Soit A un alphabet et F l'ensemble des suites de longueur n à valeur dans A. La distance de Hamming entre deux éléments a et b de F est le nombre d'éléments de l'ensemble des images de a qui diffèrent de celle de b.
Exemple : Considérons les nombres binaires suivants :

a = (0 0 0 1 1 1) et b = (1 1 0 1 0 1) alors d = 3.
La distance entre a et b est égale à 3 car il y a 3 bits diffèrents.

La valeur centrale d'un ensemble d'observations A est appelé centre médian. Considérons l'exemple donné dans les tables suivantes (voir tableau 3.3) dans lesquelles on a codé trois ensembles E1, E2 et E3 de 6 observations représentant une variable qualitative à 3, 4 et 5 modalités. Nous remarquons que la valeur médiane dans le cas du codage additif représente souvent une des modalités de la variable qualitative prise en considération.
On s'intéresse dans une première partie, aux variables qualitatives dont ses composantes ne prennent qu'un nombre limité de valeurs. En effet, les ensembles de données binaires sont intéressants et utiles pour diverses raisons. Ils présentent la forme la plus simple des données disponibles dans un or-

Variable qualitative	Codage Additif
Petite	1000
Moyenne	0100
Grande	0010
Très grande	0001

TABLE 3.4 – Exemple : la variable taille codée en codage additif.

dinateur et peuvent être utilisées pour représenter des données catégorielles. Pour traiter les données qualitatives, des méthodes statistiques ont été appliqués au cours des années et qui sont basés sur les réseaux de neurones. On trouve aussi des méthodes de classification automatique qui se reposent sur la notion de la métrique ou de probabilité. Parmi lesquels on trouve Cazals [**?**] qui propose une méthode de classification hiérarchique ascendante et Chavent [**?**] qui propose des méthodes de classification hiérarchiques descendantes. Cette méthode et ses améliorations ont utilisé la distance euclidienne qui n'est pas adaptée au traitement de ce genre de variables. C'est pourquoi nous allons chercher à utiliser une distance plus adéquate dans cette partie de nos expériences afin de valider notre algorithme CL-Ant.

On s'intéressera aux variables qualitatives ayant plusieurs modalités. Ce type de données est utilisé par exemple dans les enquêtes et les sondages dont la réponse à une question doit être unique en choisissant une seule modalité parmi les modalités proposées. En statistiques, on distingue deux manières de coder de telles observations : le codage disjonctif complet et le codage additif selon que la variable qualitative est ordinale ou disjonctive. On s'intéresse au codage additif dans cette partie expérimentation.

En d'autres termes, on représente le questionnaire de l'enquête, par exemple, sous forme d'un vecteur binaire et on code chacune de ses composantes sachant que chaque composante est une variable qualitative à plusieurs modalités. Ce codage est différent selon le type de la variable qualitative. Si elle est ordinale, c'est-à-dire ses modalités sont menées par un ordre total implicite (exemple d'une taille : petite, moyenne, grande, très grande), la variable se transforme en un vecteur binaire par le biais d'un codage binaire additif. En effet, la qème modalité de cette variable se représente suivant un vecteur de dimension m (nombre de modalités) dans lequel les q premières composantes sont égaux à 1 et les composantes restantes sont égales à 0.
Prenons l'exemple de la taille, on peut le coder comme le suivant (voir la tableau 3.4) :
Les mesures de similarités dans un espace de données binaires sont ap-

	1	X	0
1	a		b
Y			
0	c		d

TABLE 3.5 – Table de contingence. a et d représentent le nombre de fois que les deux individus choisissent la même modalité «1» ou «0». b et c représentent le nombre de fois que le premier individu (deuxième individu) choisit la modalité «1» et le deuxième individu (le premier individu) choisit la modalité «0».

pelées les distances binaires. Elles permettent de mesurer la ressemblance entre deux observations X et Y appartenant au même espace. Ceci est fait au moyen d'une table de contingence des deux vecteurs binaires associés. C'est une matrice représentant le nombre d'occurrences de 1 et de 0 et le nombre de désaccords d'individus ayant choisi les différents couples de modalités 1 et 0 (voir le tableau 3.5).
En littérature, on trouve plusieurs moyens pour calculer les indices de similarités à partir de la table de contingence. Citons à ce propos, la distnace de Hamming notée par H représente le nombre des différences entre deux vecteurs de données binaires X et Y :

$$H(X,Y) = b + c \tag{3.11}$$

3.3.3 Résultats

Les figures 3.6, 3.7 et 3.8 représentent les résultats obtenus pour les tests que nous avons réalisé. A partir de ces informations, nous pouvons effectuer plusieurs constatations. Nous testons notre méthode avec différents valeurs de α et γ (voir le tableau 3.2).

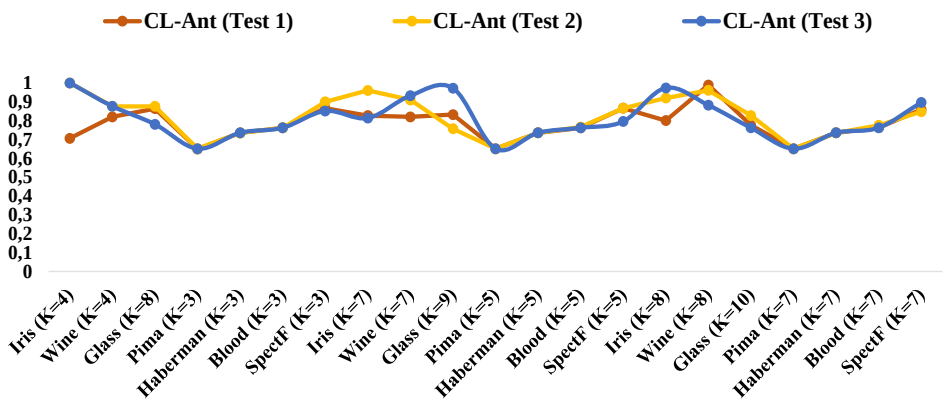


FIGURE 3.6 – Les résultats de l'indice de pureté obtenu avec CL-Ant (Test 1, Test 2, Test 3) sur les bases de données numériques après 1000 itérations.

Dans la Figure 3.6, nous obtenons une valeur de pureté la plus supérieure est égale à 1 pour la base de donnée Iris avec K = 4, où CL-Ant (Test 3) fonctionne bien, à l'exception de la base de donnée Wine avec K = 4. On remarque que les résultats CL-Ant avec trois tests étaient comparables : ils sont à peu près égale pour certaines bases de données et assez proches les uns des autres avec d'autres bases de données. Le meilleur résultat en terme d'indice de pureté est obtenu en majorité par CL-Ant (Test 3) avec α = 0.03 et γ = 0.001 comparé à CL-Ant (Test 1) et CL-Ant (Test 2) sur le même nombre de bases de données numériques.

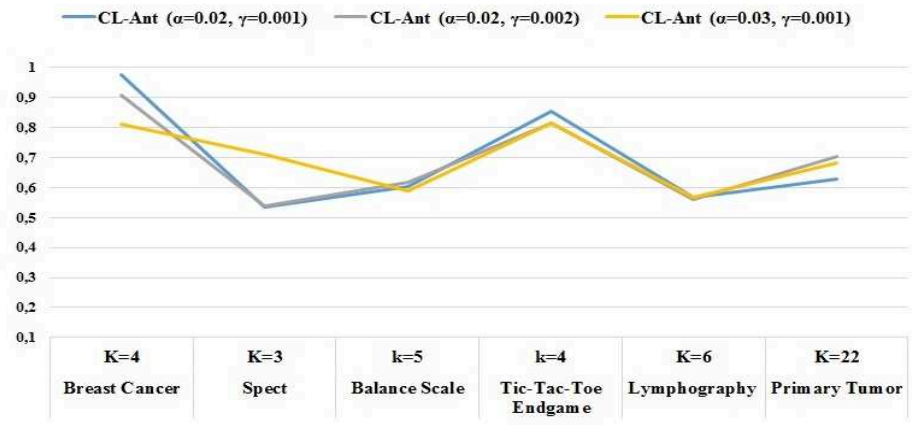


FIGURE 3.7 – Les résultats de l'indice de pureté obtenu avec CL-Ant (Test 1, Test 2, Test 3) sur les bases de données binaires après 1000 itérations.

Par contre pour les bases de données binaires, nous constatons das la figure 3.7 que la pureté obtenue par CL-Ant (Test 1) avec α = 0.02 et γ = 0.001 sont globalement similaires à ceux obtenus par CL-Ant (Test 2) et CL-Ant (Test 3) pour les deux bases de données balance scale et lymphography. On obtient une valeur de pureté importante est égale à 0.9755 pour la base de données breast cancer avec K = 4 après 1000 itérations. Le meilleur résultat est obtenu en majorité par CL-Ant (Test 1) avec α = 0.02 et γ = 0.001 pour les bases de données binaires en comparant avec les deux autres tests de CL-Ant.

Dans la Figure 3.8, nous présentons l'indice de pureté en fonction du nombre

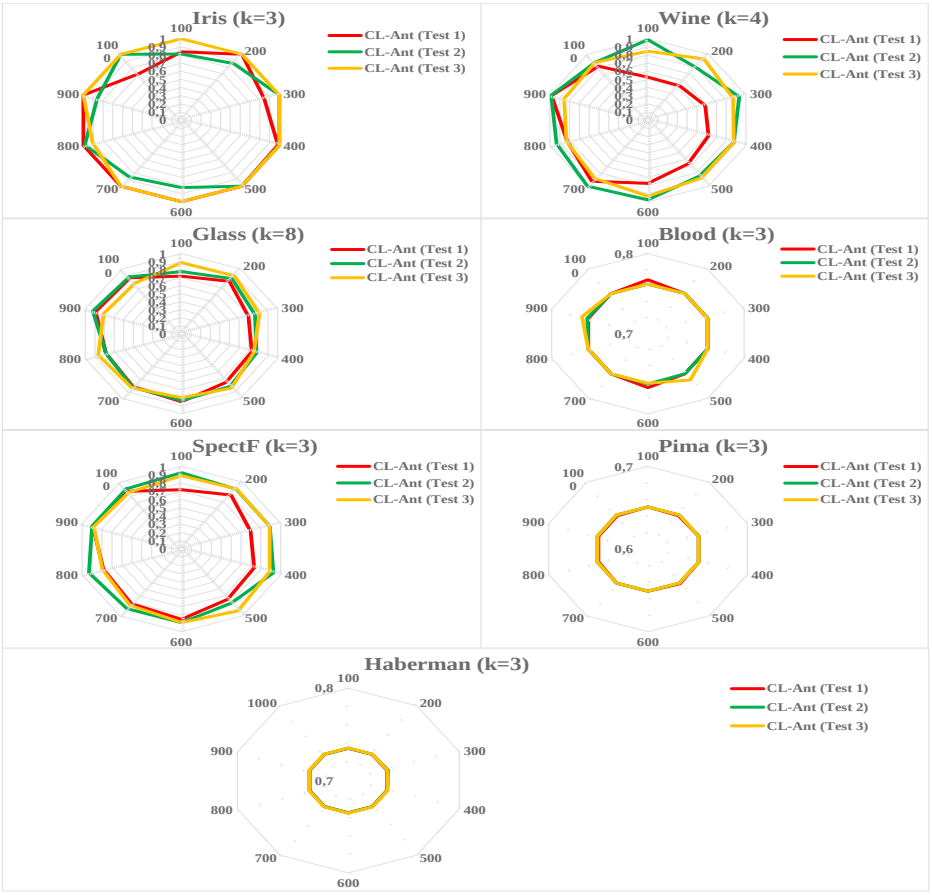


FIGURE 3.8 – Les résultats de l'indice de pureté obtenu avec CL-Ant (Test 1, Test 2, Test 3) en fonction du nombre d'itérations.

d'itérations sur différentes bases de données numériques et les tests effectués par l'algorithme CL-Ant avec les différentes valeurs de α et γ (Test 1, Test 2, Test 3). CL-Ant (Test 2) et CL-Ant (Test 3) a produit les meilleurs résultats sur la base de donnée Wine avec K = 4 et Iris avec k = 3 au cours des 1000

itérations produisant un peu mieux que CL-Ant (Test 1). Nous constatons que cette méthode fonctionne bien avec plusieurs ensembles de données du monde réel à travers les expériences.

3.4 Visualisation des résultats

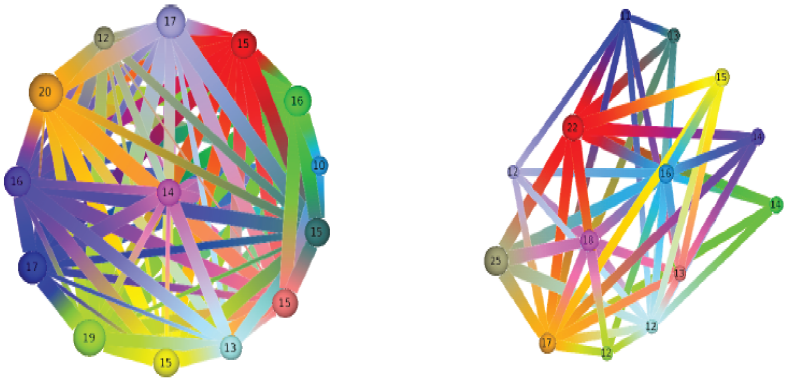
Notre approche simplifie la visualisation dans une structure de graphe dynamique, non seulement par la réduction des liens entre clusters/noeuds, mais aussi en fournissant un regroupement pertinent des structures cohérentes de classification qui peuvent être analyser par un expert de domaine. En outre les graphes dynamiques fournis par notre algorithme possèdent un grand intérêt et devraient être étudiée dans une analyse plus approfondie comme une perspective de notre approche.
Les deux paramètres α et γ influent sur les connexions entre les noeuds qui représentent la relation de proximité entre clusters. Une fois le chemin de similarité maximale représentant le taux de phéromone a été explorée, une fourmi s'intègre à un cluster CL_j selon un seuil de similarité qui dépend de la similitude des données formant ce groupe. Si $\alpha = 0$ et $\gamma = 0$, le graphe sera dans ce cas, un graphe complet dont les arêtes représentent les relations de voisinage entre les clusters selon une mesure de similarité face au type de données. Si $\alpha \neq 0$ et $\gamma \neq 0$, alors f_i assignera ou pas à des clusters voisins et le graphe résultat sera un graphe dynamique. Par conséquent, il est important de tester les différentes valeurs de ces deux paramètres.

Nous testons ainsi notre méthode CL-Ant en utilisant différentes valeurs de taux de phéromone incrémental et les valeurs d'évaporation, respectivement désignés par α et γ tel que mentionné précédemment. Ces valeurs sont calculées comme suit : Test 1, Test 2 et Test 3 correspond respectivement à $\{\alpha = 0.01 ; \gamma = 0.001\}$, $\{\alpha = 0.02 ; \gamma = 0.002\}$ et $\{\alpha = 0.03 ; \gamma = 0.001\}$. Étant donné un graphe de proximité, on définit la longueur entre deux noeuds voisins en fonction de la distance réelle (similarité) entre les deux clusters correspondants. Ensuite, tous les noeuds sont dispersés de façon aléatoire sur un plan 2D. Plusieurs exemples de résultats sont donnés dans cette section. Une fois que le graphe est visualisé, l'expert est en mesure d'explorer. Il peut détecter la structure globale du graphe, et plus précisément, les clusters existants, leur taille et leur densité, ou les relations entre ces groupes (qui cluster est près de l'autre).
Notre approche est une métaheuristique permet de visualiser des graphes de proximité à l'aide d'une plateforme Tulip [Auber 2002]. Son principal avantage est qu'elle fournit une structure non hiérarchisée. Cette fonctionnalité

simplifie l'exploration de données en offrant une visualisation conviviale. Cette visualisation permet la navigation à travers le graphe qui est présenté d'une manière non hiérarchique : l'utilisateur visualise les principaux noeuds avec un aperçu de la structure graphique, il peut alors se concentrer sur un noeud qui contient des données homogènes, et il peut également connaître l'évolution de la taille de cluster.

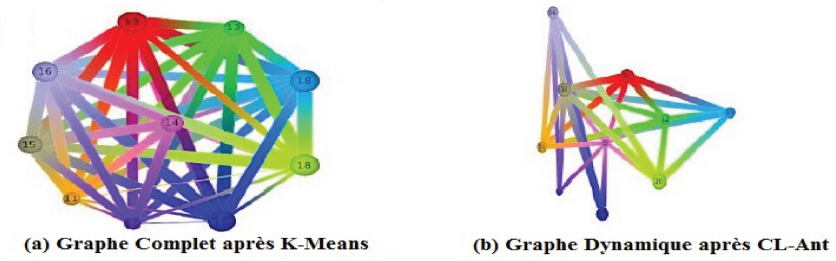
Nous utilisons Tulip [Auber 2002] comme un outil de visualisation de graphes. Afin d'analyser la connexion entre une paire de noeuds, nous avons introduit différentes visualisations illustrées dans les figures 3.9, 3.10 et 3.11. A cet effet, nous explorons la structure du graphe en analysant les arêtes entre une paire de noeuds dans le graphe résultant. Dans la visualisation de graphes, on doit manipuler plusieurs attributs au même temps : on visualise les noms, les tailles de fichiers et les types des attributs par des noeuds du graphe. Tulip nous permet de stocker un nombre important d'attributs et de modifier dynamiquement l'ensemble des attributs attachés à un graphe. On peut aussi stocker plusieurs dessins de graphe au même temps.

Dans les figures 3.9, 3.10, 3.11, 3.13 et 3.12, chaque cluster et les fourmis correspondantes ont été représentés par un noeud portant une couleur similaire. Chaque arête est colorée de manière dégradée entre deux couleurs de chaque cluster. Une fourmi f_i avec une couleur c appartenant à un cluster CL_j choisit d'emprunter une arête en fonction de phéromones. Les graphes résultants par l'algorithme CL-Ant après 100 itérations sont représentés par les figures 3.9, 3.10 et 3.11. Le but de l'algorithme de CL-Ant est de réduire le graphe initial en éliminant les arêtes avec un faible taux de phéromone et on garde les arêtes ayant un taux de phéromones élevé. Nous montrons l'évolution du graphe après différents étapes de clustering avec CL-Ant. Figures 3.13 et 3.12 (a) présentent le graphe complet obtenu après l'étape d'initialisation (exécution de l'algorithme K-Means). Figures 3.13 et 3.12 (b) montrent le graphe dynamique obtenu après l'algorithme CL-Ant pendant 100 itérations.



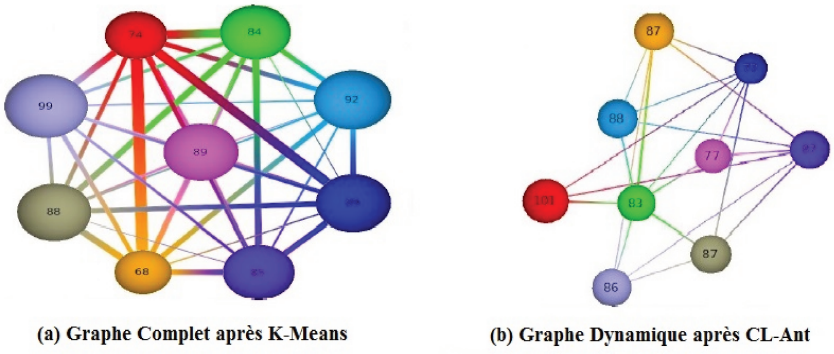
(a) Graphe Complet après K-Means (b) Graphe Dynamique après CL-Ant

FIGURE 3.9 – Exemple de graphe de proximité après l'algorithme CL-Ant pour la base de données Glass avec k = 14.



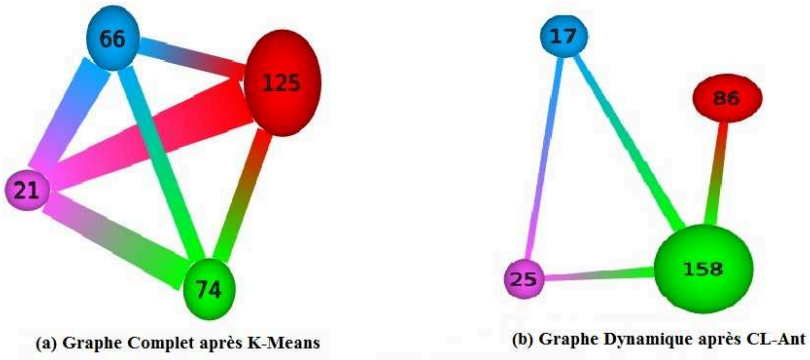
(a) Graphe Complet après K-Means (b) Graphe Dynamique après CL-Ant

FIGURE 3.10 – Exemple de graphe de proximité obtenu par l'algorithme CL-Ant pour la base de données Iris avec k = 10.



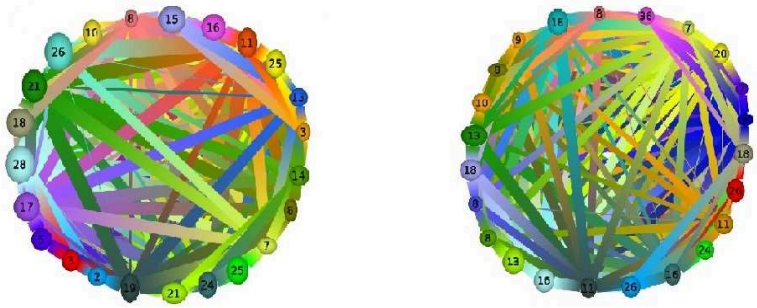
(a) Graphe Complet après K-Means (b) Graphe Dynamique après CL-Ant

FIGURE 3.11 – Exemple de graphe de proximité obtenu par l'algorithme CL-Ant pour la base de données Pima avec k = 9.



(a) Graphe Complet après K-Means (b) Graphe Dynamique après CL-Ant

FIGURE 3.12 – Exemple de graphe de proximité obtenu par l'algorithme CL-Ant pour la base de données breast cancer avec k = 4, α = 0.02 et γ = 0.001.



(a) Graphe Complet après K-Means (b) Graphe Dynamique après CL-Ant

FIGURE 3.13 – Exemple de graphe de proximité obtenu par l'algorithme CL-Ant pour la base de données primary Tumor avec k = 23, α = 0.02 et γ = 0.001.

3.5 Conclusion

Dans ce chapitre nous avons décrit une nouvelle approche inspiré du principes de reconnaissance chimique chez les fourmis réelles en particulier la construction d'une odeur coloniale commune dans un nid. Dans un premier temps, un résumé des connaissances actuelles dans le domaine de la reconnaissances chimique et de l'identification coloniale dans les sociétes de fourmis a été proposé. En étudiant et en analysant ces informations, un modèle a pu être mis en place dans un second temps. En s'appuyant sur les comportements biologiques déja connus et observés par les spécialistes de domaine. Une implémentation informatique de cet algorithme nommé «CL-Ant» a donc être menée pour résoudre le problème de classification non hiérarchique non supervisée.

L'intérêt principal de notre approche est qu'elle ne dépend pas du type de donnée à classer ; il suffit uniquement de définir une fonction de similarité pour ces nouvelles données. En effet nous avons testé notre algorithme sur des données numériques, catégorielles (binaires). Une comparaison plus tard dans le chapitre 5 sera présentée sur des données numériques avec d'autres méthodes classiques de classification. Cependant, le modèle possède des limites comme le choix de l'initialisation par l'algorithme K-Means, les valeurs de mise à jour de taux de phéromones α et γ.
Dans la chapitre suivant, nous introduisons une version incrémentale de l'algorithme appelé CL-AntInc, qui s'adapte à la venue de nouvelle donnée sans refaire tout le calcul permettant ainsi de gérer des données en flux.

CHAPITRE 4

EXTENSION DE L'ALGORITHME CL-ANT : VERSION INCRÉMENTALE

Sommaire

4.1 Introduction 120
4.2 Rappel des principes généraux de notre modèle de fourmis 120
4.3 Description de l'algorithme CL-AntInc 121
4.3.1 Principes 121
4.3.2 Choix de l'échantillon initial 123
4.3.3 Les phases de classification 124
4.4 Etude expérimentale et validation 125
4.4.1 Jeux de données utilisés 125
4.4.2 Résultats 128
4.5 Visualisation 132
4.6 Conclusion 133

Résumé

Nous décrivons dans ce chapitre l'extension que nous avons pu apporter à CL-Ant. Nous présentons une version incrémentale, CL-AntInc offrant ainsi une possibilité de manipuler un volume de données important, une évolution de l'algorithme CL-Ant. Bien qu'il soit proche de ce dernier nous allons étudier, dans cette version, la capacité d'une nouvelle espèce de fourmi de reconnaître ses congénères pour changer l'identité coloniale avec les membres de l'espèce partenaires dans la colonie (section 4.2).
Nous présentons donc cet algoritjme, nommé CL-AntInc. Pour cette version les expérimentations sont réalisées sur des bases de données numériques et binaires pour évaluer les performances de notre méthode (section 4.4). Nous représentons également l'aspect visuel des graphes générés avec notre méthode dans la section 4.5.
Nous concluons avec une présentation des avantages et des inconvénients de notre approche (section 4.6).

4.1 Introduction

Pour une nouvelle version de l'algorithme nous utilisons les mêmes principes et notations définis précédemment dans le chapitre 3 : les fourmis f_1,..., f_n représentent chacune une donnée de la base et sont placées initialement dans les clusters CL_j. Ensuite, nous simulons successivement une action pour chaque fourmi. Une fourmi peut avoir deux états : soit elle se déplace, soit elle est affectée à un cluster. Pour les fourmis en déplacement, les actions effectuées vont dépendre de la similartité de la fourmi et le seuil d'acceptation du cluster vers lequel elle se déplace. Les fourmis ne perçoivent la structure que localement. Pour une fourmi f_i en déplacement et affectée dans les clusters $CL_{j'}$ située à la structure, le voisinage perceptible par f_i est limité aux clusters voisins et aux fourmis filles du cluster.

4.2 Rappel des principes généraux de notre modèle de fourmis

Dans cette nouvelle version, les mêmes principes de comportement des fourmis réelles sont appliqués comme dans le chapitre précédent.
Notre approche se fonde également sur l'odeur coloniale collective construite dans le même nid et la capacité à former des colonies mixtes en intégrant les

fourmis appartenant à des espèces différentes. Cela permet d'étudier la capacité d'une nouvelle espèce de fourmis de reconnaître et de mettre en évidence les congénères des changements dans l'identité coloniale avec les membres de l'espèce partenaires dans la colonie [**?**] [**?**]. Dans le modèle artificiel que nous proposons, chaque fourmi représente une donnée initialement située dans un cluster, chaque nid représente un cluster. Les fourmis se déplacent d'un cluster à un autre, pour trouver le nid ou le cluster qui les intègre le mieux. La colonie va évoluer en intégrant de nouveaux nids (nouveaux clusters). La colonie est représentée par un graphe dynamique illustré dans la figure 4.1. Chaque nid (cluster) est représenté par un noeud dans le graphe. Le graphe dynamique évolue à chaque fois qu'un nouveau cluster est ajouté (nouveaux noeuds, ajouter les arêtes) pour obtenir le meilleur partitionnement des données.

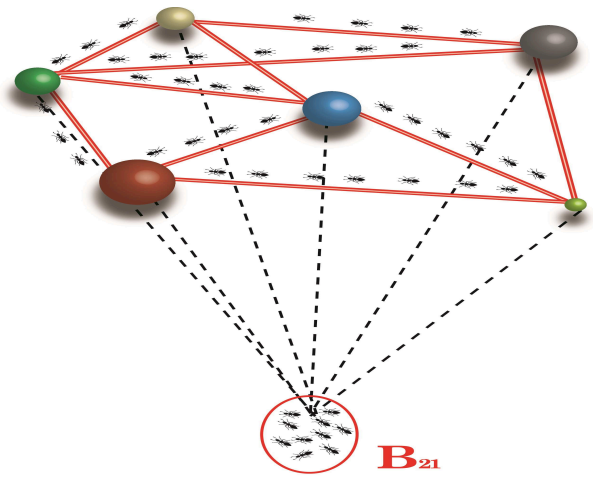


FIGURE 4.1 – Étape incrémentale : découper l'ensemble de données (pour simuler le flux) en N_B blocs.

4.3 Description de l'algorithme CL-AntInc

4.3.1 Principes

Nous présentons une extension nommé CL-AntInc [Algorithme 4] qui détermine comment classer les nouveaux objets sans besoin à reclasser la base des objets initiaux. L'objectif d'un tel algorithme est alors de classer les données et modifier les classes existantes, sans besoin à traiter de nouveau l'ensemble des données disponibles. Ces principes rendent ces méthodes efficaces sur

de grandes bases de données. Par conséquent, nous proposons une version incrémentale pour traiter les flux de données. Ce nouvel algorithme incrémental permet la construction de graphes de proximité pour un ensemble de données important afin de résoudre le problème de clustering. Il était inspiré des odeurs coloniales et le processus de renforcement à base des phéromones observés chez les fourmis réelles où les fourmis deviennent progressivement des membres d'un groupe, puis rejoignent successivement d'autres clusters plus similaires. Chaque fourmi artificielle représente une donnée. La façon dont les fourmis se déplacent dans une structure de graphe dépend de la similarité entre les données.

Nous considérons plusieurs blocs de tailles aléatoires classés de façon consécutive selon les données générées à partir des bases de données. Chaque bloc est ensuite classé sous une forme non-hiérarchique dans un graphe résultant de l'algorithme CL-Ant. En utilisant un premier échantillon de données B_1 sera initialement classé selon les mêmes principes observés dans le chapitre 3. Au départ, toutes les fourmis sont positionnées en clusters et chaque fourmi se déplace en essayant de s'intégrer aux clusters voisins suivant les règles de comportement définis dans l'algorithme CL-Ant (section 3.3 du chapitre 3). Ensuite, le principe de clustering est de classer les données restantes progressivement. Une fois la phase de pré-classification est complétée selon les étapes précédentes, les données seront sélectionnées par blocs de taille complètement aléatoire. Elles seront classées dans le graphe généré par l'algorithme CL-Ant tout en appliquant les principes suivants : une fourmi f_i est associée à chaque donnée puis les blocs $B_{2i'}$ sont classés successivement l'un après l'autre en fonction de leur arrivée ; Simuler le déplacement d'une fourmi à un cluster jusqu'à son intégration (voir la figure 4.1) .
Nous définissons le seuil d'un bloc comme suit :
– $Th_{B2i'}$: chaque bloc $B_{2i'}$ représente un nid qui possède un seuil d'affiliation.

$$Th_{B2i'} = (\sum_{i=1}^{n} \sum_{j=1, i\neq j}^{n} (\| x_{f_i} - x_{f_j} \|^2)/n) + \min(\| x_{f_i} - x_{f_j} \|^2)/2 \quad (4.1)$$

Où f_i et $f_j \in B_{2i'}$, x_{f_i} est le vecteur de données associé à la fourmi f_i.

Une fourmi f_i appartenant à un bloc sera ensuite tenter de joindre les autres fourmis situées dans les anciens clusters si ($Sim(f_i, W_{B_i}) < Th_{B_i}$) (où i $\in$ {2,..., N_B}) et W_{B_i} est le barycentre de B_i) . De la même manière que dans CL-Ant, f_i est suffisamment similaire aux autres fourmis «soeurs» appartenant au cluster $CL_{j'}$ si Sim(f_i, $W_{j'}$) $>= Th_{CL_{j'}}$) la fourmi f_i est affectée

au cluster $CL_{j'}$, sinon nous allons f_i à un autre groupe voisin. Dans le cas où le f_in'est pas acceptée par l'ancien cluster, elle gardera son emplacement d'origine, ce qui conduit à la possibilité de créer de nouvelles classes qui peuvent apparaître au fil du temps. Chaque bloc représente un nouveau nid sera supprimer lorsque toutes les fourmis sont affectées.

Algorithme 4 Une vue génnérique de l'algorithme de classification incrémentale CL-AntInc.

1: **Entrée** : B, X : des données non structurées où $B \leftarrow X \cup B_i$
2: **Sortie** : $G_{proximityInc} \leftarrow$ CL-AntInc(B)
3: **Initialisation :**
   – N_B : le nombre total de blocs.
   – G', Gi, G_1, $G_{proximityInc}$: graphes résultants après l'exécution des étapes de l'algorithme
4: **Partitionnement** $B = \bigcup_{i=1}^{N_B} B_i$
   /* $CL - Ant_1$ exécuter l'algorithme 2 */
5: $G_1 \leftarrow$ CL-Ant$_1(B_1)$
6: **Pour** i = 2 à N_B {
7: G' $\leftarrow$ Addition (B_i à G_{i-1})
   /* CL-Ant$_2$ exécute l'algorithme 2 à partir de l'instruction 4 */
8: $Gi \leftarrow$ CL-Ant$_2(G')$
9: }**Fin Pour**
10: $G_{proximityInc} \leftarrow G_{N_B}$

4.3.2 Choix de l'échantillon initial

Le choix de l'échantillon s'effectue à partir de la base initiale, sa taille est limitée à 7 000 données en fonction du nombre total des données formant la base à tester, taille maximale pouvant être tolérée pour l'instant par l'algorithme utilisé pour classer l'échantillon par les deux premiers étapes de l'agorithme CL-AntInc. Pour cela, dans la phase d'initialisation par K-Means et la première phase de classification par l'algorithme CL-Ant, nous choisirons par exemple un échantillon dont la taille est aléatoire. Ensuite dans la phase suivante nous l'appliquerons à des données arrivant en flux. Pour nos tests, nous avons choisi dans un premier temps d'extraire aléatoirement un échantillon contenant des données appartenant à plusieurs classes dont le but d'inialiser la structure du graphe pour que les fourmis peuvent se déplacer. Dans un deuxième temps, nous avons voulu tester la capacité de CL-AntInc à créer de nouveaux clusters non présents dans l'étape précédente par l'arrivée des blocs de données en flux. Dans cette phase CL-AntInc récupère les don-

nées arrivant en bloc dont la taille est complètement aléatoire. Nous avons choisi des bases de données réelles pour tester et valider notre algorithme.

4.3.3 Les phases de classification

Nous présentons dans cette section une version incrémentale de notre méthode de classification respectant les principes définis dans la section 3.4 du chapitre 3. Ce nouvel algorithme s'appuie sur la version CL-Ant décrite dans le chapitre précédent. Elle a l'avantage de ne posséder aucun paramétrage rendant ainsi sa mise en oeuvre plus facile. L'objectif de cet algorithme noté CL-AntInc est de classer les données sans modifier au besoin les classes préexistantes, sans traiter à nouveau l'ensemble des données disponibles. Nous montrerons dans la section 4.3.3 que ce principe rend CL-AntInc efficace pour un grand volume de données.

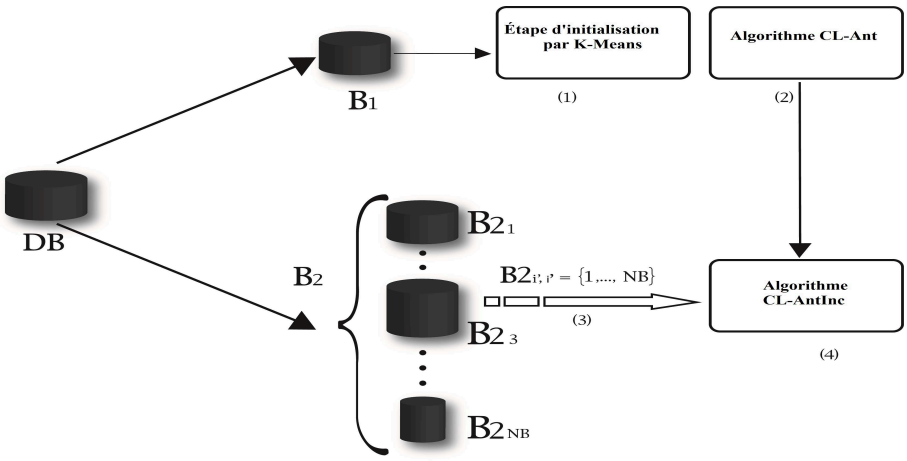


FIGURE 4.2 – Principe général de l'algorithme CL-AntInc.

L'algorithme CL-AntInc est divisé en quatre étapes principales (voir la figure 4.2) :

1. **Étape d'initialisation en utilisant l'algorithme K-Means** CL-Ant$_1$() (voir la figure 4.2(1)) : la structure initiale est un graphe complet dont les noeuds sont des clusters de données connues a priori et dont les arêtes représentent les relations de voisinage entre les noeuds.
2. **Première étape de clustering** CL-Ant$_2$() (voir la figure 4.2(2)) : dans cette étape, l'algorithme CL-Ant est appliqué pour construire un graphe dynamique qui représente la préservation de la topologie des clusters de données.

3. **Découper l'ensemble de données en N_B blocs pour simuler le flux** (voir la figure 4.2(4)) : nous considérons plusieurs blocs (B_i où i $\in$ {2,..., N_B}) avec une taille aléatoire comme un ensemble de données consécutives générées à partir d'une base de données. Les fourmis artificielles se déplacent vers le nid qui possède la même odeur coloniale. Chaque bloc représente un nouveau nid à supprimer lorsque toutes ses fourmis seront affectées. Les blocs de données sont construit à partir d'un échantillonnage de taille aléatoire. Une fourmi f_i est associée à chaque donnée ; les fourmis appartenant aux blocs B_i sont classés successivement l'un après l'autre en fonction de leur arrivée ; l'attribution d'une fourmi à un cluster a été simulé jusqu'à son intégration dans le nouveau cluster.

4. **Étape incrémentale** (voir la figure 4.2(3)) : dans cette étape, nous introduisons l'étape incrémentale où une nouvelle espèce de fourmis intègre de nouvelles groupes et détermine de nouvelles relations de voisinage en utilisant la structure du graphe dynamique. Lorsque les fourmis sont placées dans un nouveau cluster, elles se déplacent afin de trouver un autre endroit où les fourmis possèdent une odeur coloniale similaire en utilisant les mêmes règles de comportement présentées dans le chapitre 3. La navigation à travers le graphe est présenté d'une manière non hiérarchique : l'utilisateur visualise les principaux noeuds avec un aperçu de la structure du graphe, il peut alors se concentrer sur le noeud qui contient des données homogènes, et il peut aussi connaître l'évolution de la taille des clusters.au cours des étapes de l'algorithme CL-AntInc.

   Une fois l'étape 3 est terminée, les données en blocs de tailles différentes arrivent un par un. Tous les blocs sont classés selon un graphe dynamique généré à l'étape 2. Les fourmis des blocs sont ensuite déplacés vers un autre cluster en utilisant le chemin qui a la plus forte concentration de phéromones suivant les règles de l'algorithme CL-AntInc [Algorithme 4].

4.4 Etude expérimentale et validation

4.4.1 Jeux de données utilisés

Afin de tester les capacités de CL-AntInc, nous utilisons différentes bases de données extraites de la Machine Learning Repository [**?**] dont les caractéristiques générales sont résumées dans le tableau 4.1 : Nb est le nombre total de données formant les bases de données, N_{Att} est le nombre d'attri-

Datasets	Nb	K	N_{Att}	B_1	N_B	C_R	C_F
Statlog	6435	3	37	700	4	7	7
SpamBase	4601	3	57	300	4	2	7
WaveForm	5000	3	40	600	4	3	7
MagicGamma	19020	3	11	1600	4	2	7
StatlogShuttle	58000	3	9	2000	10	7	13
WineQuality	4898	3	12	600	6	8	9
Car Evaluation	1728	3	6	100	3	4	6
Congressional Voting Records	435	4	16	1000	6	2	10
Chess	3196	3	36	500	3	2	6
Connect-4	67557	4	42	7000	7	3	11
Neursery	12960	5	8	5000	7	5	12

TABLE 4.1 – Description des bases de données utilisées par l'algorithme CL-AntInc (Machine Learning Repository).

but, K le nombre de clusters théorique déterminé par l'algorithme K-Means à l'étape d'initialisation, B_1 est le nombre des données du bloc initial, N_B est le nombre total des blocs formant de nouvelles groupes à intégrer, C_R le nombre de classes réelles et C_F représente le nombre de classes trouvées.

Bases de données		B_i(Taille des blocs)	
	Test 1	**Test 2**	**Test 3**
SpamBase	{1201 ;735 ; 1421 ;944}	{1462 ;1149 ; 634 ;1056}	{1841 ;114 ; 252 ;2144}
WineQuality	{857 ;1762 ;512 ; 115 ;202 ;1150}	{980 ;1127 ;62 ; 235 ;257 ;1819}	{993 ;954 ;558 ; 480 ;257 ;1257}
WaveForm	{34 ;1856 ; 1009 ;1701}	{1701 ;1671 ; 574 ;1284}	{924 ;610 ; 325 ;2741}
Statlog(Heart)	{1619 ;1117 ; 1047 ;2325}	{1213 ;2176 ; 935 ;1711}	{2689 ;1617 ; 702 ;1027}
MagicGamma	{6360 ;2836 ; 1679 ;5145}	{6789 ;2434 ; 3380 ;3417}	{4309 ;3420 ; 417 ;7874}
Statlog(Shuttle)	{15197 ;335 ;6713 ; 7531 ;3697 ;9627}	{17263 ;6065 ;51 ; 353 ;1118 ;17650}	{19200 ;2168 ;4790 ; 3465 ;5285 ;7592}

TABLE 4.2 – Les bases de données numériques utilisées dans nos tests par l'algorithme CL-AntInc, où B_i est la taille différente des blocs choisie d'une manière aléatoire.

Datasets		B_i(Size of blocks)	
	Test 1	**Test 2**	**Test 3**
Congressional Voting Records	{105 ;56 ;124}	{120 ;31 ;184}	{86 ;22 ;227}
Chess	{944 ;323 ;183 ; 24 ;346 ;376}	{374 ;755 ;290 ; 182 ;111 ;484}	{605 ;462 ;99 ; 196 ;343 ;491}
Car Evaluation	{204 ;302 ;722}	{29 ;159 ;1040}	{156 ;133 ;939}
Connect-4	{1452 ;18113 ; 9664 ;10704 ; 4155 ;2661 ; 13808}	{28976 ;11298 ; 5753 ;5260 ; 1783 ;3736 ; 3751}	{18839 ;11196 ; 10484 ;5753 ; 1760 ;2043 ; 10482 ;}
Neursery	{1853 ;405 ;2291 ; 1627 ;813 ;280 ; 691}	{3807 ;1152 ;1071 ; 769 ;558 ;235 ; 368}	{168 ;3668 ;1589 ; 585 ;793 ;50 ; 1107}

TABLE 4.3 – La taille des différents blocs obtenus par l'algorithme CL-AntInc sur des données binaires.

Nous présentons les différentes caractéristiques des blocs de données générés pour les bases de données binaires (voir le tableau 4.3) et les données numériques (voir tableau 4.2) : B_i où i $\in$ {2,..., N_B} est la taille aléatoire d'un ensemble de données consécutives générée à partir d'une base de données testée.
Il est plus fréquent de calculer la similarité entre deux objets en utilisant une mesure de distance définie en fonction des données en entrée. Depuis la similarité est fondamentale pour déterminer un cluster, la mesure de distance doit être choisi avec soin. Le choix de la distance est très important pour la méthode de classification en fonction du type de données [Jain 1988]. Dans notre approche, nous utilisons la distance de Hamming qui est adaptée à notre algorithme lorsque l'ensemble de données sélectionnés est binaire et nous appliquons la distance euclidienne lorsque les bases de données testées sont numériques.

Nous avons évalué notre algorithme sur 5 bases de données binaires, dont le nombre de données varie entre Nb = 435 et 67557 exemples et 6 bases de données numériques avec des tailles différentes comme illustré dans le tableau 4.1.

4.4.2 Résultats

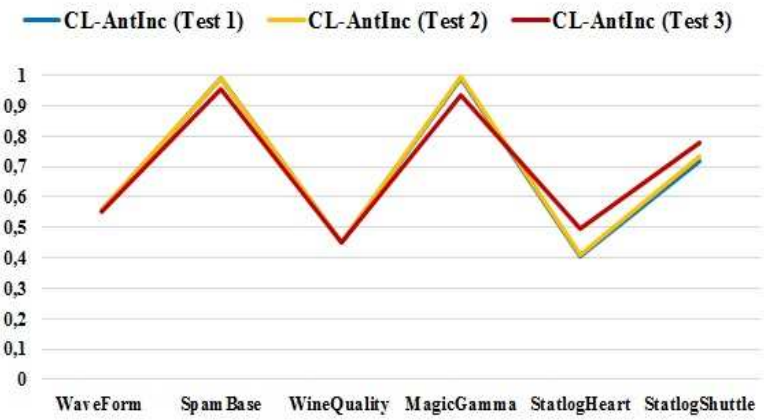


FIGURE 4.3 – Les résultats de l'indice de pureté obtenu avec CL-AntInc (Test 1, Test 2, Test 3) sur les bases de données numériques après 1000 itérations.

Nous constatons que cette méthode fonctionne bien avec les bases de données réelles à travers les expériences. L'algorithme CL-AntInc a deux paramètres α et γ, ces derniers ont une influence sur les connexions entre les clusters. Nous testons notre méthode en utilisant différentes valeurs de taux de phéromones et d'évaporation de phéromones, respectivement désignés par α et γ tel que mentionné précédemment dans le chapitre 3.

Ces valeurs sont calculées comme suit : Test 1, Test 2 et Test 3 correspond respectivement à $\{\alpha = 0.01 ; \gamma = 0.001\}$, $\{\alpha = 0.02 ; \gamma = 0.002\}$ et $\{\alpha = 0.03 ; \gamma = 0.001\}$.
Pour 7 classes trouvés et 2 classes réelles avec une valeur de pureté 0.991 (voir Figure 4.3). Il peut également expliquer l'algorithme de mauvaises performances dans le test 1 avec $\alpha = 0.01$ et $\gamma = 0.001$ à d'autres ensembles de données, il est faible pour la base de données WinQuality par exemple dont la pureté est égale 0.453. CL-AntInc (Test 2) avec $\alpha = 0.02$ et $\gamma = 0.002$ exécute mieux pour MagicGamma ensemble de données où K = 3, N_B = 4, B_1 = 3000 données et B_i = (6789, 2434, 3380, 3417) est la taille de différents blocs. et il mieux performant que les deux autres tests de CL-AntInc avec la pureté est égale à 0.9942 et la valeur de Rand à 0.66.
Nous remarquons que l'algorithme CL-AntInc est moins performant pour quelques bases de données comme StatlogShuttle, par exemple, dont l'indice de pureté est égale à 0.4 et 0.6 correspond à l'indice de Rand (voir figure 4.4).
Dans la figure 4.5, nous utilisons l'algorithme CL-AntInc (Test 2) avec α = 0.02 et γ = 0.002 pour présenter l'indice de pureté testé sur trois ensembles

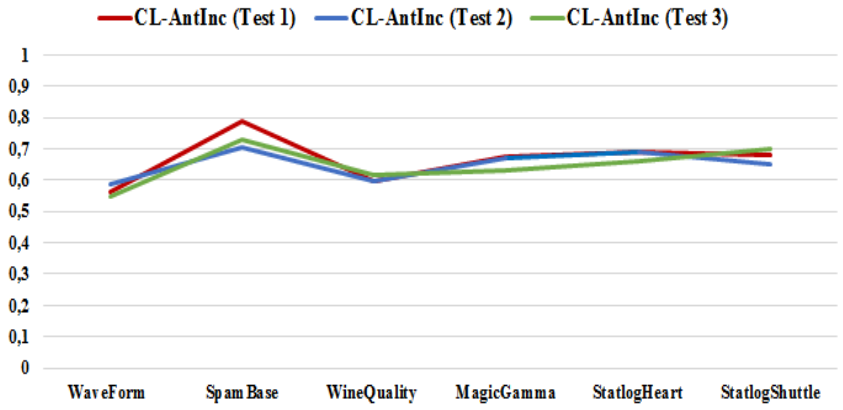


FIGURE 4.4 – Les résultats de l'indice de Rand obtenu avec CL-AntInc (Test 1, Test 2, Test 3) sur les bases de données numériques après 1000 itérations.

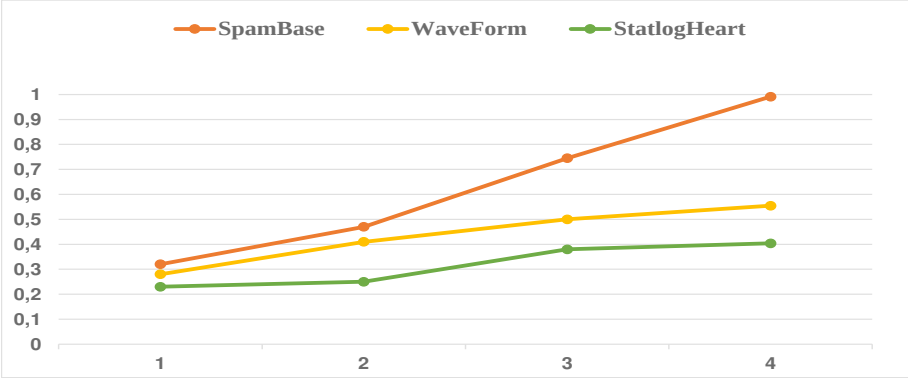


FIGURE 4.5 – Évolution de l'indice de pureté obtenu avec CL-AntInc (Test 2) après l'intégration de N_B blocs appliquée aux différents bases de données numériques.

de données (SpamBase, WaveForm, StatlogHeart) avec N_B = 4. Nous remarquons que les valeurs de pureté sont augmentées après l'intégration de chaque bloc.

Nous représentons la fréquence de sortie des fourmis noté $freq_S$ en pourcentage (voir l'équation 4.2) et nous calculons l'indice de pureté après l'intégration de blocs de données testés sur trois bases de données (Statlog, Statlog-Shuttle, WinQuality) (voir la figure 4.6).

$$freq_S = \frac{N_{f_i}}{T_{f_n}} \times 100 \tag{4.2}$$

Avec N_{fi} est le nombre des sorties de la fourmi f_i appartenant au bloc B_i, T_{f_n} est le nombre totale des fourmis et i $\in$ 1, ...N_B.

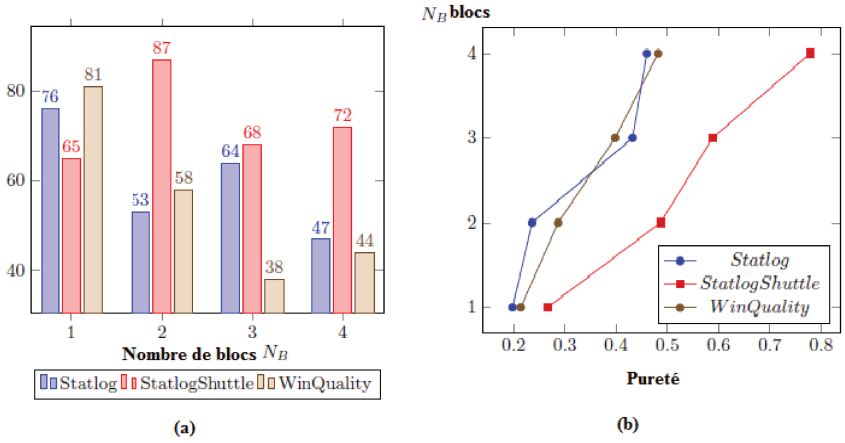


FIGURE 4.6 – (a) : Le pourcentage de fréquence de sortie des fourmis. (b) : L'évolution de l'indice de pureté obtenu par CL-AntInc (Test 1) après l'intégration de N_B blocs appliqué à différents bases de données.

Après l'intégration du deuxième bloc, la fréquence de sortie des fourmis est de 87% avec une pureté est égale à 0.5 pour la base de données StatlogShuttle. Nous remarquons que la pureté est faible en comparant avec un pourcentage de sortie fourmis élevé pour les bases de données Statlog et WinQuality. Ceci explique que les fourmis quittent leur cluster d'origine, mais elles ne sont pas affectées à un cluster voisin, ceux-ci représente une limite de l'algorithme CL-AntInc.

Nous présentons pour toutes les bases de données binaires [1] les valeurs de pureté dans la figure 4.7, les résultats obtenus par CL-AntInc (Test 1) avec α = 0.02 et γ = 0.001 sont semblables à ceux obtenu par CL-AntInc (Test 2) et CL-AntInc (Test 3) pour la base de donnée Congressional Voting Records. L'algorithme CL-AntInc (Test 1) avec α = 0.02 et γ = 0.001 a donné les meilleurs résultats par rapport à CL-AntInc (Test 1) et CL-AntInc (Test 2) pour les autres base de données testées. Nous avons obtenu une valeur de pureté supérieure égale à 0.914 pour la base de données Congressional Voting Records après 1000 itérations où K = 3, NB = 3, B_1 = 100 données et B_i = (105, 56, 124) est les différents blocs. La pureté est faible pour ma base de données Connect-4 dont la pureté est égale à 0.48, car le nombre d'attributs

1. Nesrine Masmoudi, Hanane Azzag, Mustapha Lebbah, Cyrille Bertelle, Maher Ben Jemaa : CL-AntInc Algorithm for Clustering Binary Data Streams Using the Ants Behavior. The 20^{th} International Conference on Knowledge-Based and Intelligent Information and Engineering Systems Kes, York-United Kingdom, (2016)

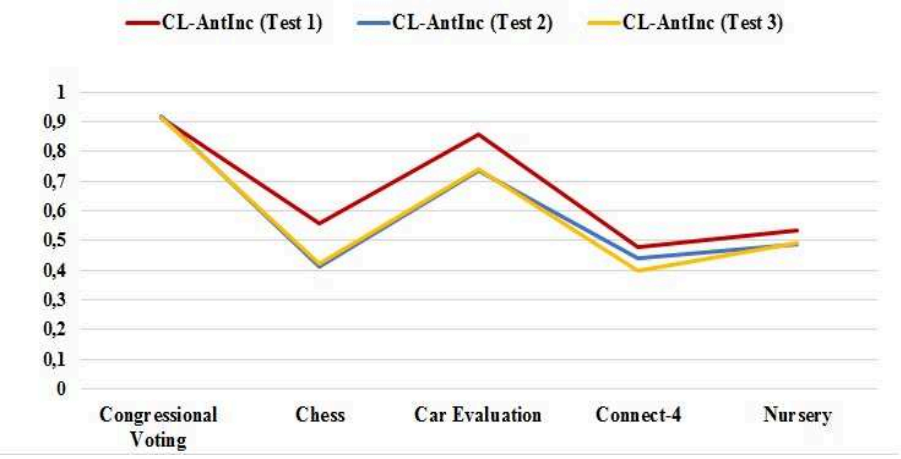


FIGURE 4.7 – Les résultats de l'indice de pureté obtenu avec l'algorithme CL-AntInc (Test 1, Test 2, Test 3) sur les bases de données binaires après 1000 itérations.

est plus important. Les résultats de l'algorithme CL-AntInc avec ses trois tests étaient comparables : l'égalité de l'indice de pureté pour quelques bases de données, et l'un est plus performant que l'autre pour les autres bases de données.

4.5 Visualisation

Le but de l'algorithme est de réduire le graphe initial en éliminant les arêtes avec un taux de phéromones bas et on maintient les arêtes avec un taux de phéromones élevé. La figure 4.9 montre les différents graphes testés sur la base de données Statlog(Heart) où K = 6, N_B = 7, B_1 = 500 données et B_i = (698, 424, 159, 437, 439, 265, 1513). En outre, nous montrons l'évolution du graphe après les différentes étapes de classification avec l'algorithme CL-AntInc. Nous considérons maintenant l'aspect visuel de l'algorithme pour la construction d'un graphe de proximité. Lorsque toutes les fourmis sont testées et affectées, on obtient un graphe dynamique où chaque noeud est représenté par un «cluster». Nous utilisons l'outil «Tulip» [Auber 2002] comme une plateforme pour visualiser et analyser le graphe résultat. Afin d'analyser la connexion entre une paire de noeuds, nous introduisons les différentes visualisations illustrées aux figures 4.8, 4.9 et 4.10 pour des exemples de bases de données binaires et numériques. Les figures 4.8, 4.9 et 4.10 (a) présentent le graphe complet obtenu après l'étape d'initialisation (exécution par K-Means). Les figures 4.8, 4.9 et 4.10 (b) montrent le graphe dynamique obtenu après l'algorithme CL-Ant pendant 100 itérations. Les figures 4.8, 4.9 et 4.10 (c) montrent le graphe obtenu après l'introduction d'un bloc de données (nouveau nid ou cluster). Les figures 4.8, 4.9 et 4.10 (d) représentent le résultat final de l'algorithme incrémental CL-AntInc une fois que tous les blocs sont mises en place.

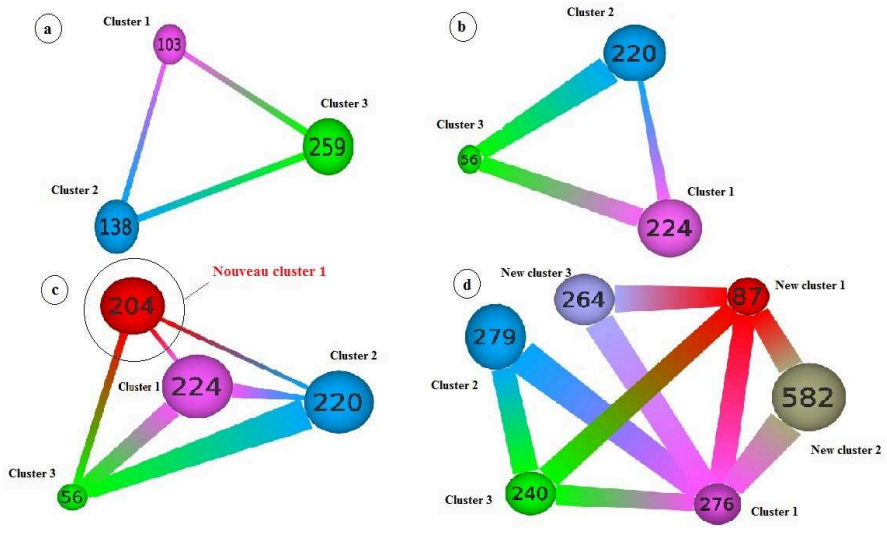


FIGURE 4.8 – Évolution du graphe obtenu à partir du l'algorithme CL-AntInc (Test 1) appliqué à la base de données binaire Car Evaluation avec K = 3.

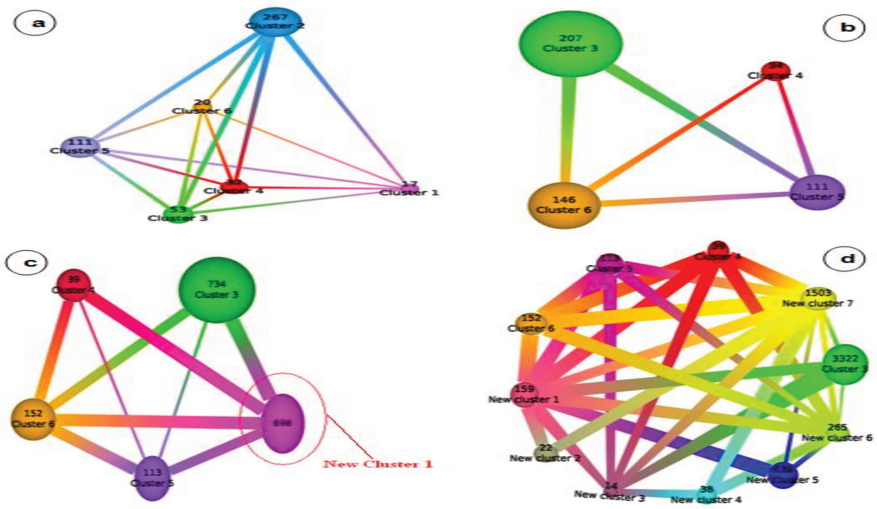


FIGURE 4.9 – Évolution du graphe obtenu à partir du l'algorithme CL-AntInc (Test 1) appliqué à la base de données numérique Statlog(Heart) avec K = 6.

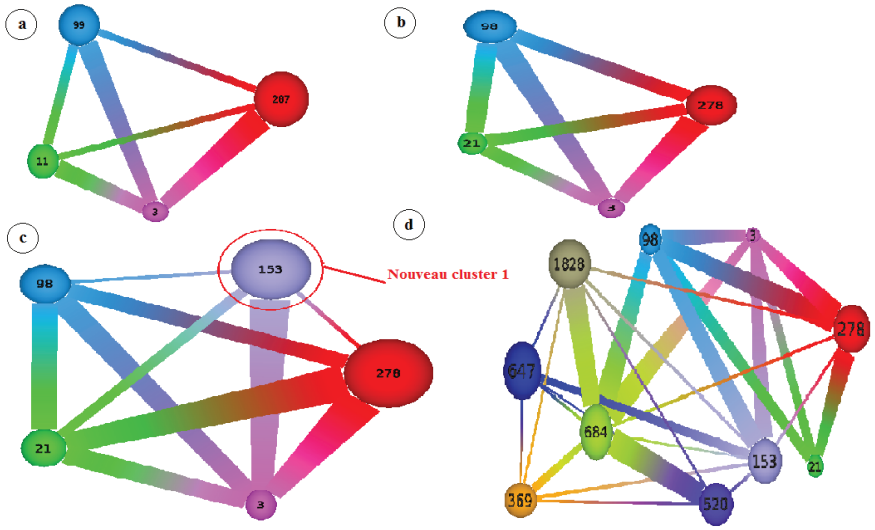


FIGURE 4.10 – Évolution du graphe obtenu à partir du l'algorithme CL-AntInc (Test 3) appliqué à la base de données numérique SpamBase avec K = 4.

4.6 Conclusion

Nous avons présenté dans ce chapitre une extension pouvant être apportée à CL-Ant. Nous avons développé et testé une version incrémentale appelée CL-AntInc, ne possédant aucun paramètre. Ce dernier offre également des possibilités d'incrémentalité et de ce fait nous avons développé CL-AntInc pour l'exécution sur un volume de données important.

Nous avons testé et validé notre algorithme sur des bases de données réelles numériques et binaires. Nous allons comparer dans le chapitre suivant l'algorithme CL-Ant à d'autres algorithmes classiques de classification : la CAH (Classification Ascendante Hiérarchique), les K-Means (voir le chapitre 1). Nous comparons également la version incrémentale avec d'autres algorithmes de clustering en flux.

CHAPITRE 5

ÉTUDE COMPARATIVE

Sommaire

5.1 Introduction 136
5.2 Algorithmes étudiés : comparaison avec des méthodes traditionnelles 136
5.2.1 Expérimentations 137
5.2.2 Comparaison avec les méthodes classiques 141
5.2.3 Temps d'exécution 141
5.3 Comparaison avec des méthodes de flux 143
5.3.1 Expérimentations 143
5.3.2 Étude comparative par les données numériques . . . 143
5.3.3 Temps d'exécution 145
5.4 Étude de la Complexité 146
5.5 Propriétés de l'algorithme 148
5.6 Conclusion 149

Résumé

Nous comparons dans ce chapitre les deux algorithmes de fourmis artificielles définis dans les chapitres précédents (CL-Ant et CL-AntInc) en les mettant en concurrence avec une méthode de classification hiérarchique (CAH), une méthode de partitionnement classique (K-Means) et trois méthodes incrémentaux (D-Stream, DenStream et CluStream).
Nous présentons les résultats obtenus sur les jeux de données réelles que nous avons utilisés pour nos précédentes évaluations (sections 5.3 et 5.4).
Nous étudions la complexité des deux algorithmes proposés (section 5.5).
Enfin nous concluons par énumérer les propriétés de nos modèles en montrant que les fourmis sont une source pertinente pour résoudre le problème de classification non hiérarchique non supervisée et les résultats obtenus sont encourageants (section 5.6 et 5.7).

5.1 Introduction

Afin de déterminer l'efficacité relative des différents algorithmes présentés dans les chapitres 3 et 4, nous avons utilisé la même méthodologie de tests qui nous a permis d'étudier les paramètres de ces deux algorithmes (voir section 3.4 du chapitre 3). Nous comparons ensuite ces résultats à ceux obtenus par des méthodes traditionnelles tel que la CAH, les K-Means, et des méthodes incrémentaux tel que D-Stream [Tu 2009], DenStream [Cao 2006], CluStream [Aggarwal 2003] et ClusTree [**?**]. Nous avons réalisé nos tests sur les 22 bases numériques et catégorielles définies dans les chapitres précédents.
Nous avons initialisé les paramètres de nos algorithmes de fourmis (CL-Ant, CL-AntInc) aux valeurs définies dans les chapitres précédents et pour lesquels les algorithmes associés apportent les meilleurs résultats.

5.2 Algorithmes étudiés : comparaison avec des méthodes traditionnelles

Pour la conception de la CAH nous utilisons le critère de Average linkage comme distance d'agrégation. Mais étant donné que les bases testées sont toutes numériques, elle est réalisée de la même manière que celle définie dans le chapitre 2.

Pour la méthode des K-Means, le principal inconvénient est qu'il faut lui fournir une partition de départ de K classes. Pour cela, le plus simple est d'en générer une aléatoirement avec différents valeurs de K. Avec une hypothèse que nous fournissons aux K-Means un nombre de classes de départ proche du nombre de classes originelles.

5.2.1 Expérimentations

Nous avons réuni les résultats obtenus dans les figures : 5.1, 5.2, 5.3, 5.4, 5.5 et 5.6. Nous avons réalisé nos tests sur les 14 bases numériques définies dans les chapitres précédents. Comme nous l'avons indiqué dans les résultats présentés dans les chapitres 3 et 4, notre évaluation se base sur la pureté et sur l'indice de Rand et l'indice de NMI pour chaque méthode testée. Ces résultats correspondent à 1000 itérations au maximum de chaque algorithme proposés. Nous avons réalisé plusieurs études comparatives sur des ensembles de données classiques du UCI Repository of Machine Learning. Les ensembles de données que nous avons testés sont de type numérique.

5.2.1.1 Données numériques

- **L'indice de pureté**

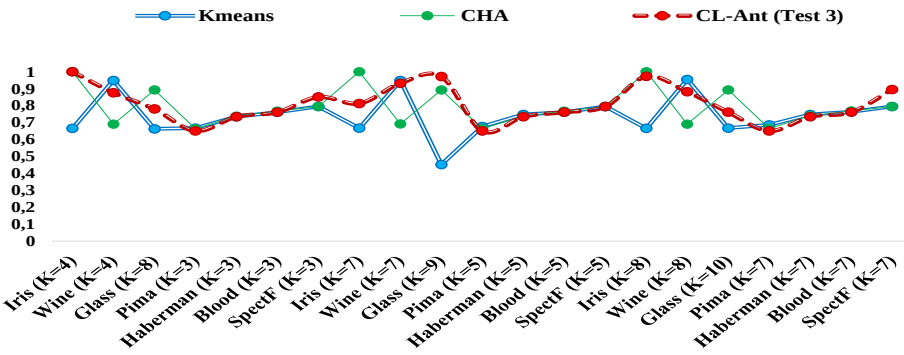


FIGURE 5.1 – Les résultats de l'indice de pureté obtenus avec CL-Ant (Test 3 : α = 0.03 et γ = 0.001), K-Means et CAH sur les bases de données numériques après 1000 itérations.

Nous comparons notre algorithme pour deux méthodes classiques et bien connus dans la classification de données, à savoir CAH et K-Means. Nous avons testé notre méthode avec différents α et γ. A partir des résultats obtenus sur l'indice de pureté (voir les figures 5.1 et 5.2), on peut voir que

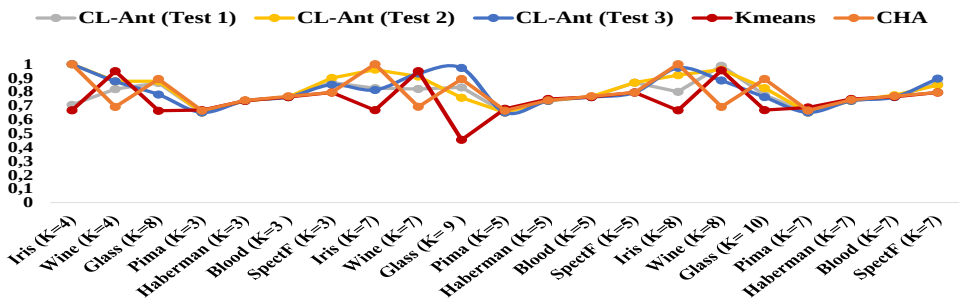


FIGURE 5.2 – Les résultats de l'indice de pureté obtenus avec CL-Ant (Test 1, Test 2, Test 3), K-Means et CAH sur les bases de données numériques après 1000 itérations.

la CAH et CL-Ant obtiennent les meilleurs résultats de pureté sur le même nombre des bases de données. En effet, un partitionnement plus important entraîne un petit effectif dans chaque classe trouvée avec des données très similaires entre elles, ce qui limite les possibilités d'erreur. On peut considérer cela comme un biais de la mesure de pureté qui, à l'extrême, vaut 100% si chaque donnée forme une classe. Il faut donc idéalement combiner l'information de pureté avec le nombre de classes trouvées. Il faut cependant noter que le nombre de classes réelles n'est pas nécessairement représentatif du nombre de groupes que l'on trouve dans les données. Pour certaines données, telles que Iris ou Wine, il y a trois classes réelles et à l'aide de l'indice de pureté, par contre, pour des données comme Pima (voir la figure 5.2), il y a deux classes réelles (ces données sont connues pour être très bruitées) la valeur de pureté est constant de 0.651 pour les différent valeurs de K et pour les trois tests de CL-Ant. Si l'on observe le comportement des algorithmes pour ces bases en ce qui concerne le nombre de classes et la pureté, CL-Ant (Test 3) est meilleure dans tous les cas (voir la figure 5.1) pour les données Iris, et pourtant on sait très bien qu'il existe trois «groupes» au sein des données Iris (CL-Ant (Test 3) trouve une valeur de pureté égale à 0.9733, K-Means possède 0.66 avec différent valeur de K). On remarque que notre algorithme CL-Ant qui est une hybride dans le sens où la recherche du nombre de classes est effectuée à l'initialisation par un algorithme classique en classification comme K-Means possède des valeurs de pureté suppérieur à celles obtenues par les K-Means seulement. Ces valeurs s'expliquent par le fait que lorsqu'on obtient un très grand nombre de classes, la probabilité d'avoir une bonne pureté augmente, surtout si ces classes nombreuses sont homogènes.

- **L'indice de Rand**

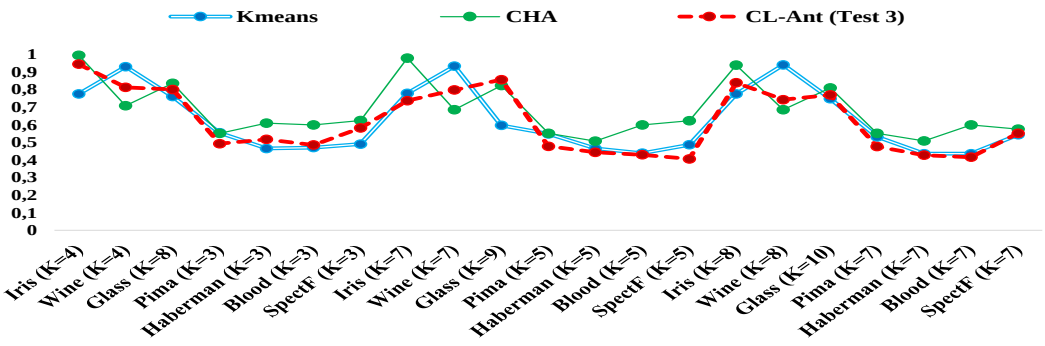


FIGURE 5.3 – Les résultats de l'indice de Rand obtenus avec CL-Ant (Test 3 : α = 0.03 et γ = 0.001), K-Means et CAH sur les bases de données numériques après 1000 itérations.

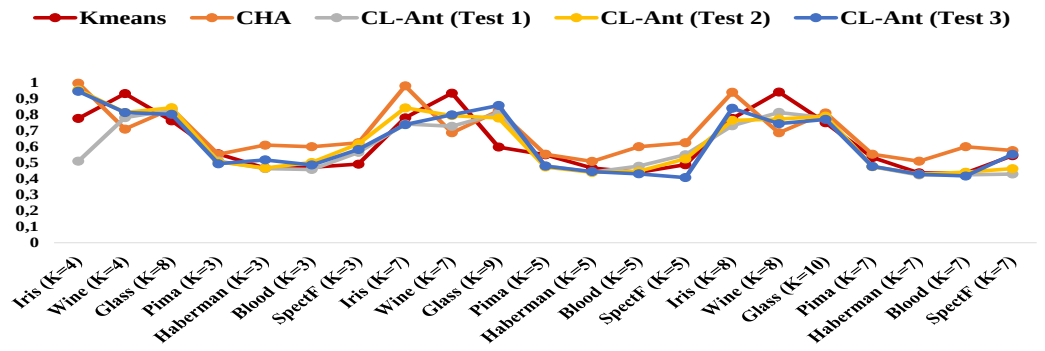


FIGURE 5.4 – Les résultats de l'indice de Rand obtenus avec CL-Ant (Test 1, Test 2, Test 3), K-Means et CAH sur les bases de données numériques après 1000 itérations.

Globalement K-Means trouve une valeur de Rand plus faible que les autres méthodes (voir les figures 5.3 et 5.4), ce qui, compte tenu le bases de données comportant peu de K classes à l'étape d'itialisation de notre algorithme, peut lui procurer un avantage d'avoir un Rand plus élevé. Notons néanmoins que CL-Ant obtient également les meilleurs résultats pour 3 bases sur 7, on le retrouve aussi en seconde position pour la reste des bases de données testées. Les moins bons résultats sont observés pour CL-Ant (Test 1) et CL-Ant (Test 2). Pour CAH, en moyennant, la meme valeur de Rand que CL-Ant.

On peut expliquer la mauvaise performance de l'algorithme K-Means par l'initialisation à une grande valeur du paramètre K (nombre de classes de la partition initiale). C'est pour cela on a fusionné l'algorithme CL-Ant à ce dernier pour améliorer la qualité de partitionnement. On peut également remarquer que CL-Ant génère un nombre de classes trouvées plus important avec une bonne qulaité de classification.

- **L'indice de NMI**

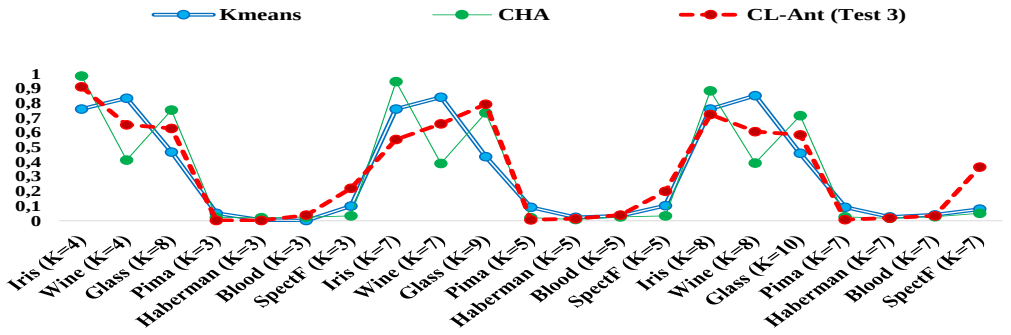


FIGURE 5.5 – Les résultats de l'indice NMI obtenus avec CL-Ant (Test 3 : $\alpha = 0.03$ et $\gamma = 0.001$), K-Means et CAH sur les bases de données numériques après 1000 itérations.

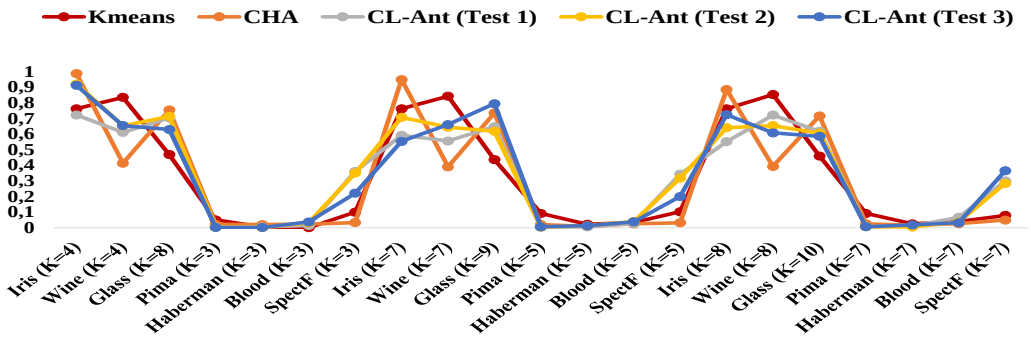


FIGURE 5.6 – Les résultats de l'indice NMI obtenus avec CL-Ant (Test 1, Test 2, Test 3), K-Means et CAH sur les bases de données numériques après 1000 itérations.

En ce qui concerne la valeur de NMI (voir les figures 5.5 et 5.6), les meilleures valeurs sont obtenues par CL-Ant (Test 3) qui est le premier pour 3 bases sur

7 surtout si le choix de la valeur de K est proche du nombre de la classe réelles (Classe réelle +1). CAH obtient également de bonnes performances avec le deuxième meilleur résultat pour la moitié des bases. On peut remarquer que des bases sont systématiquement plus difficiles que les autres. C'est le cas par exemple de Glass, Pima, blood et Haberman. Ces bases correspondent à des cas ou les classes sont mal séparées les unes des autres. C'est pour cette raison par exemple que la CAH est systématiquement dépassée par CL-Ant. Pour CL-Ant (Test 3), bien qu'il ait trouvé une nombre de classes assez important par rapport aux autres tests de l'algorithme, les résultats obtenus sur la NMI ne sont pas aussi bons que ceux des K-Means.

5.2.2 Comparaison avec les méthodes classiques

Pour les données numériques, au vu des résultats obtenus dans les figures concernant la pureté, le Rand et l'indice de NMI, les meilleurs résultats sont obtenu par la CAH suivi de près par CL-Ant (Test 3), puis CL-Ant (Test 2), CL-Ant (Test 1) et enfin K-Means. De la même manière que nous l'avons expliqué précédemment, on considère ce fait comme un biais de la mesure de pureté. Néanmoins on peut souligner que la différence entre les écarts de pureté de notre algorithme de fourmis et CAH reste relativement faibles.

Il faut donc analyser conjointement le nombre de classes trouvées et la pureté. Le but est alors de choisir parmi les différentes tests de notre algorithme celle qui va minimiser le nombre de classes trouvées sans dégrader la pureté. C'est pourquoi il est préférable de conclure sur l'efficacité de notre méthode en examinant les figures précédentes. On peut voir alors que CAH et CL-Ant (Test 3) apparaissent comme étant les deux meilleurs algorithmes. Ensuite CL-Ant (Test 2) et CL-Ant (Test 1) obtiennent pratiquement les mêmes résultats. Enfin notre algorithme ainsi que la CAH surclassent K-Means.

En analysant ainsi les indices d'évaluation on se rend compte que finalement les résultats obtenus par CAH sont plus proches de ceux de CL-Ant (Test 3) contrairement à ce qui avait été observé pour les résultats de K-Means puisque notre méthode améliore la qulaité de classification pour la rendre approximaient ceux de CAH.

5.2.3 Temps d'exécution

Nous présentons dans cette section une étude sur les temps d'exécution obtenus par CL-Ant avec les trois tests sur les données binaires et les données numériques. Les valeurs obtenues sont présentées dans les figures 5.7 et 5.8.

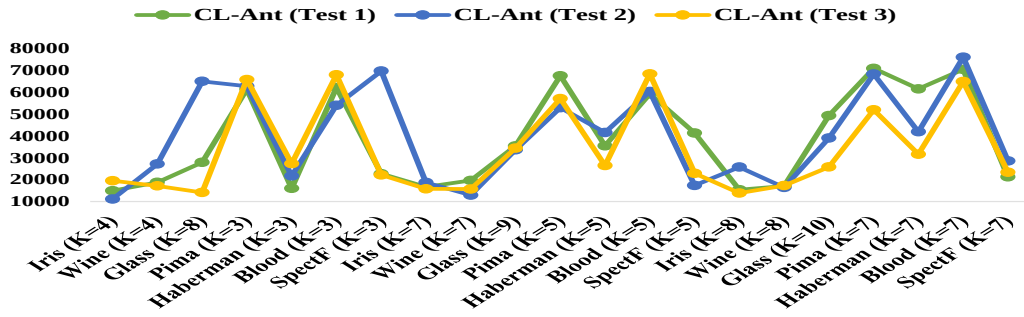


FIGURE 5.7 – Le temps d'exécution en millisecondes obtenus avec CL-Ant (Test 1, Test 2, Test 3) sur des base de données numériques après 1000 itérations. L'algorithme a été programmé en Java et les tests ont été effectués sur une machine avec un processeur Pentium 52.6 GHz et 512 Mo de RAM.

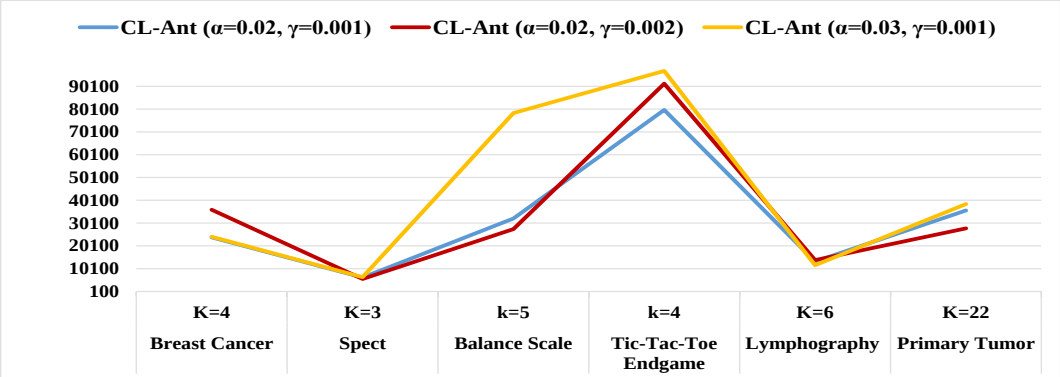


FIGURE 5.8 – Le temps d'exécution en millisecondes obtenus avec CL-Ant (Test 1, Test 2, Test 3) sur des bases de données binaires après 1000 iterations. L'algorithme a été programmé en Java et les tests ont été effectués sur une machine avec un processeur Pentium 52.6 GHz et 512 Mo de RAM.

Pour chaque base ces valeurs prennent en compte uniquement le temps nécessaire au déroulement de l'algorithme de classification incluant ainsi l'initialisation des paramètres pour notre algorithme de fourmis. Par exemple l'étape d'initialisation, le calcul de la similarité, le calcul des seuils de similarité présents pour chaque cluster K.
L'extraction des données et le calcul des similarités entre les données de chaque base sont communes à toutes les méthodes. Les temps correspondant sont présentés respectivement dans les figures 5.7 et 5.8 pour les données nu-

mériques et binaires. Il est difficile de conclure sur les bases pour lesquelles ces opérations s'exécutent en un temps très faible. En effet, les variations de temps introduites par le système d'exploitation (occupation du disque, gestion des caches) rendent ces informations extrêmement bruitées. Nous pouvons néanmoins conclure que pour les bases de moins de 1000 données les algorithmes d'extraction et de calcul de similarité s'exécutent très rapidement.
En ce qui concerne les temps d'exécutions, les meilleurs résultats sont obtenus par CL-Ant (Test 1) avec α = 0.02 et γ = 0.001, ces résultats sont très rapide et comme indiqué précédemment restent bruités par les opérations effectuées par le système d'exploitation.

5.3 Comparaison avec des méthodes de flux

5.3.1 Expérimentations

Nous avons réuni dans le tableau 5.1 un récapitulatif des résultats de pureté et de Rand obtenus par notre algorithme CL-AntInc et ceux des algorithmes incrémentaux (DenStream, CluStream, D-Stream et ClusTree). Nous présentons également des études comparatives par l'indice NMI dans la figure 5.9. Étant donné que nous utilisons des jeux de tests sur les bases numériques.

5.3.2 Étude comparative par les données numériques

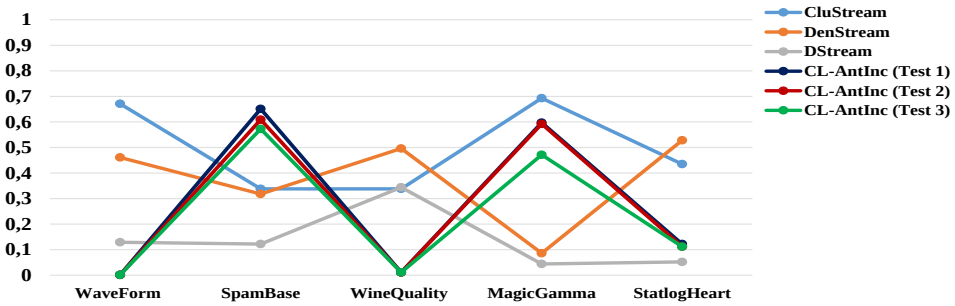


FIGURE 5.9 – Les résultats de l'indice NMI obtenus avec CL-AntInc (Test 1, Test 2, Test 3), DenStream, CluStream et D-Stream sur les bases de données numériques après 1000 itérations.

Au vu des résultats, les meilleurs valeurs pour la pureté sont obtenus par CL-AntInc (Test 2). CL-AntInc (Test 1) et CL-AntInc (Test 3) obtiennent

α	γ	Bases de données	CL-AntInc		CluStream		DenStream		D-Stream	
			Purity	Rand	Purity	Rand	Purity	Rand	Purity	Rand
0.01	0.002	WaveForm	0.5546	0.5635	0.343	0.966	0.532	0.524	0.801	0.038
		SpamBase	0.9913	0.7859	0.856	0.694	0.854	0.6934	0.827	0.632
		WineQuality	0.453	0.5956	0.424	0.874	0.684	0.898	0.495	0.618
		MagicGamma	0.9928	0.6774	0.371	0.893	0.626	0.013	0.502	0.003
		StatlogHeart	0.4039	0.6891	0.457	0,919	0.722	0.873	0.691	0.042
		StatlogShuttle	0.719	0.682	0.584	0.88	0.362	0.776	0.524	0.803
0.02	0.001	WaveForm	0.5544	0.5873	0.343	0.966	0.532	0.524	0.801	0.038
		SpamBase	0.9887	0.7031	0.856	0,694	0.854	0.6934	0.827	0.632
		WineQuality	0.4504	0.5963	0.424	0.874	0.684	0,898	0.495	0.618
		MagicGamma	0.9942	0.6687	0.371	0.893	0.626	0.013	0.502	0.003
		StatlogHeart	0.4107	0.6914	0.457	0,919	0.722	0.873	0.691	0.042
		StatlogShuttle	0.732	0.652	0.584	0.88	0.362	0.776	0.524	0.803
0.03	0.001	WaveForm	0.5522	0.5489	0.343	0.966	0.532	0.524	0.801	0.038
		SpamBase	0.9537	0.7296	0.856	0,694	0.854	0.6934	0.827	0.632
		WineQuality	0.4508	0.6163	0.424	0.874	0.684	0.898	0.495	0.618
		MagicGamma	0.9376	0.6326	0.371	0.893	0.626	0.013	0.502	0.003
		StatlogHeart	0.4966	0.6601	0,457	0,919	0.722	0.873	0.691	0.042
		StatlogShuttle	0.781	0.698	0.584	0.88	0.362	0.776	0.524	0.803

TABLE 5.1 – Les indices de Pureté et de Rand entre CL-AntInc et les différents algorithmes incrémentaux (DenStream, CluStream, D-Stream) ; Le poids sur l'arête est augmenté par la valeur α qui représente le taux de phéromone, tous les arêtes sont diminués par la valeur γ corresponds aux clusters de haute dissimilarité.

des résultats comparables et très proches. Enfin CL-AntInc est significativement meilleur que CluStream, DenStream et D-Stream pour MagicGamma ensemble de données où K = 3, N_B = 4, B_1 = données 3000. et B_i = (6789, 2434, 3380, 3417) est la taille différente des blocs et surclasse les deux autres tests de CL-AntInc aussi bien avec une pureté est égale à 0.9942.. CL-AntInc (Test 1) reste moins performant pour la base de donnée WinQuality, par exemple dont la pureté est 0.453. En ce qui concerne le nombre de classes trouvées, CL-AntInc (Test 2) reste en première position pour l'ensemble de données SpamBase avec 7 classes trouvés et 2 classes réelles avec une valeur de pureté est égale à 0.991. Cette valeur révèle que lorsque vous obtenez un grand nombre de classes, la probabilité d'avoir une pureté élevé est importante. Cette fois-ci CL-AntInc (Test 2) surclasse les autres algorithmes. Nous pouvons expliquer ces résultats par le fait que notre méthode est hybride donc relativement plus performantes que les autres algorithmes. Cette heuristique améliore significativement les résultats obtenus. Nous avons obtenu les mêmes résultats pour les deux indices de Rand et NMI quand CL-AntInc

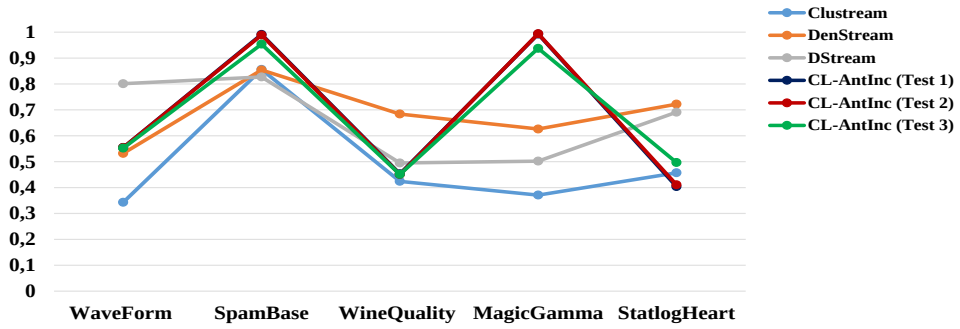


FIGURE 5.10 – Les résultats de l'indice de pureté obtenus avec CL-AntInc (Test 2), DenStream, CluStream, D-Stream et ClusTree sur les bases de données binaires après 1000 itérations.

(Test 2) fonctionne bien que les autres tests de CL-AntInc et les autres algorithmes.

5.3.3 Temps d'exécution

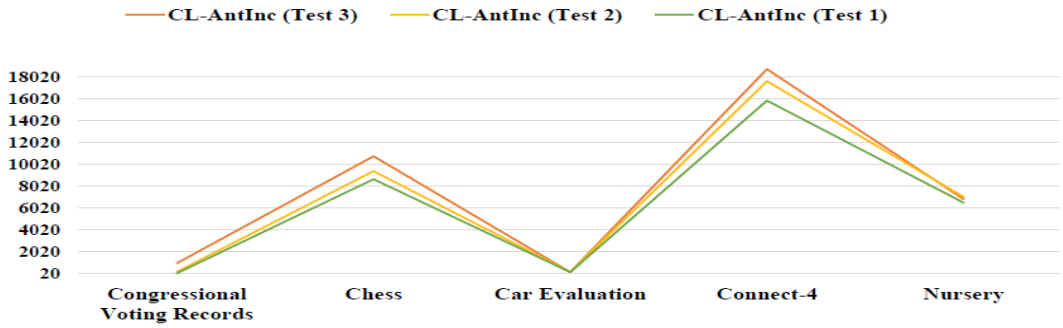


FIGURE 5.11 – Le temps de calcul total en secondes obtenu par l'algorithme CL-AntInc pour les bases de données binaires (P4, 6GB RAM, avec Java).

Nous présentons les temps d'exécution de l'algorithme CL-AntInc avec différentes valeurs α et γ testé sur les données numériques en comparant avec les méthodes incrémentaux dans les figures 5.12. Les trois tests de CL-AntInc obtiennent les meilleurs temps sur la totalité des bases testées et ils sont proche au temps d'éxécution des algorithmes incrémentaux comme DenStreamm, D-Stream et CluStream. On peut noter que pour la base de données SpamBase, elle nécessite un temps de fonctionnement plus large, car le

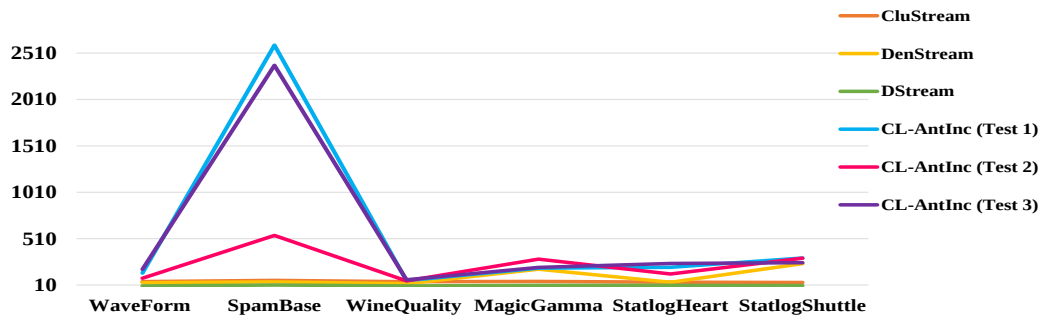


FIGURE 5.12 – Le temps d'exécution comparative en millisecondes entre l'algorithme CL-AntInc (Test 1, Test 2, Test 3) et les méthodes incrémentaux tels que DenStream, CluStream et D-Stream pour les ensembles de données.

nombre d'attributs est plus important. En ce qui concerne les temps d'exécutions de notre algorithme appliqué sur les données binaires (voir la figure 5.11), les meilleurs résultats sont obtenus par CL-AntInc (Test 1) par rapport aux autres tests qui s'exécutent en un temps faible et comme indiqué précédemment restent bruités par les opérations effectuées par le système d'exploitation. Bien que nous remarquons pour la base de données Connect-4 est plus lente que les autres bases testées cela peut être expliquer par le nombre important d'attributs constituant ces données.

5.4 Étude de la Complexité

Les algorithmes de clustering sont proposés pour résoudre le même objectif, mais leur temps de calcul [**?**] peut être différent en fonction de leurs principes. Habituellement, le temps de l'efficacité d'un algorithme en cours d'exécution est liée à la longueur du processus d'apprentissage ou le nombre de mesures de complexité (de temps) et les emplacements de stockage (complexité spatiale). La complexité est souvent désignée par O ou Θ. Dans le tableau 5.2, nous étudions la complexité temporelle des principaux algorithmes présentés précédemment [**?**].

Vu la complexité quadratique de la majorité des méthodes de classification hiérarchiques, ces dernières ne peuvent généralement pas être appliquées à de grands volumes de données. Généralement la complexité d'un algorithme ascendant est polynomiale tandis que celle d'un algorithme descendant est exponentielle. En effet, la première étape d'une méthode ascendante im-

Algorithmes de classification	Complexité
K-means	Θ(KNniter)
AntTree	Θ(NlogN)
AHC	Θ(N^3)
CL-Ant	Θ(KNniter + N^3)
CL-AntInc	Θ(KNniter + N_B(N^3))

TABLE 5.2 – Calcul de complexité des algorithmes de classification

plique d'évaluer toutes les agrégations possibles de deux objets parmi n soit $n(n-1)/2$ possibilités. Les méthodes hiérarchiques classiques sont difficilement utilisables compte tenu de leurs complexité de mémoire élevé $(O(n^2))$. D'autre part, de nombreux algorithmes de clustering ont une complexité temporelle intrinsèque élevée. Par exemple, les K-Means classique est NP-difficile c'est un algorithme rapide dont la complexité a une croissance au plus polynomiale en fonction de la taille du donnée. L'approche des algorithmes à base de populations de fourmis est sans aucun doute intéressante pour le problème de classification, mais ces méthodes sont souvent limitées à des données numériques ou encore à un choix difficile et très complexe de paramètres. Les insectes, généralement dotés de capacités restreintes lorsqu'ils sont pris séparément peuvent réaliser ensemble des tâches d'une grande complexité. A ce jour, toutes les méthodes de classification inspirées des fourmis artificielles sont utilisées pour résoudre des problèmes de partitionnement (classification hiérarchique) ou d'optimisation, et aucune n'est appliquée à la classification non hiérarchique comme une hybridation avec une heuristique pour une classification de données plane comme les K-Means dans une structure de graphe dynamique.

D'une manière générale, pour les algorithmes présentés dans les chapitres précédents la complexité est difficile d'évaluer avec précision, comme on le verra dans les résultats. Dans les figures 5.7, 5.8 et 5.12, nous présentons les temps d'exécution pour CL-AntInc qui a été implémenté en Java et exécuté sur Ubuntu 14.04 à 4 CPU avec 6 Go de RAM. L'algorithme K-Means est une méthode très rapide, sa complexité est Θ(nKNiter) où n est le nombre de données, K est le nombre de classes et Niter est le nombre d'itérations. La première étape de classification nécessite Niter itérations comme un critère d'arrêt. Le traitement de l'algorithme CL-Ant comporte trois étapes itératives : quitter le cluster source, l'afectation au cluster cible et la mise à jour des paramètres.

Pour mettre fin à la formation des N données/fourmis et sortir de la troisième boucle, il faut une complexité de N^3 opérations. Enfin, CL-Ant a une

complexité de Θ(nKNiter + Niter $\times N^3$) où la première partie représente la complexité de l'étape d'initialisation par K-Means qui est égal à (nKNiter). L'algorithme CL-AntInc nécessite des blocs pour construire NB $\times$ (K + NB) représentés les noeuds d'un graphe. L'arrivée du bloc de données, comporte trois étapes itératives : quitter le bloc, l'affectation au cluster le plus simmilaire et la mise à jour des paramètres. L'algorithme CL-AntInc utilise un sous-ensemble de données dans chaque bloc. Ces étapes nécessitent (NB $\times N^3$) opérations. Un nouveau cluster est ajouté, et cette étape est exécutée une fois par bloc. Enfin, le CL-AntInc a une complexité de Θ(nKNiter + Niter $\times N^3$).

5.5 Propriétés de l'algorithme

En comparant avec les méthodes classiques, on constate que nos algorithmes donnent des valeurs de puretés plus haute que K-Means sur la majorité des bases testées. De plus, bien que K-Means soit une méthode très rapide en terme de temps d'exécution, elle reste handicapée par le fait qu'elle ne peut classer que des données numériques. Quant à CAH, elle apporte les meilleurs résultats sur la majorité des bases testées mais reste proche en terme de pureté que notre méthode.
Pour les résultats de nos algorithmes, CL-Ant semble apporter les meilleures performances. CAH arrive en seconde position. Quant à CL-AntInc a l'avantage de ne posséder aucun paramètre à ce qui est défini dans la première version CL-Ant ce qui facilite sa mise en oeuvre son inconvénient majeur est de créer beaucoup plus de classes en comparant avec le nombre des classes réelles des bases de données testées. Néanmoins le principal avantage de nos méthodes se situe dans leur rapidité (temps d'exécution très faible), ce qui nous permet de les utiliser pour de nombreuses applications.

Le travail réalisé n'est pas pour autant achevé. En effet dans les deux versions on pourrait penser à trier les similarités des fourmis par noeud en ordre croissant, ceci nous évitera peut être à tester la totalité des fourmis/données formant le cluster. De plus, ceci optimise nos algorithmes et diminuera le temps d'exécution et la complexité de l'algorithme. On pourrait aussi hybrider notre méthode avec une autre heuristique que les K-Means qui permettrait d'initialiser la structure de départ pour éliminer la contrainte d'initialiser le nombre de classe K. Malgré cette limite on peut remarquer que notre approche d'hybridation pourrait améliorer significativement les résultats de la classification obtenue. Une méthode hybride donc relativement plus performante que les autres méthode de classification. Quant à notre méthode il optimise un par-

titionnement initialement généré par les K-Means à l'étape d'initialisation ce qui lui permet d'approximer plus efficacement l'homogèneité des classes trouvées afin d'onbtenir une bonne qualité de classification.

5.6 Conclusion

Nous avons montré dans ce chapitre que les fourmis sont une source d'inspiration pertinente pour résoudre le problème de classification non supervisée non hiérarchique à base des graphes dynamiques. Les bases de données à manipuler ne contiennent pas toutes des attributs numériques, nous avons étendu alors l'utilisation des fourmis à des données qualitatives avec un codage additif.
Dans un premier temps, nous avons comparé nos deux versions de l'algorithme à des méthodes classiques de classification non supervisée (CAH et K-Means). Dans un second temps, nous les avons comparées à des approches incrémentales qui traitent les données en flux.

CONCLUSION GÉNÉRALE

Bilan du travail réalisé

Dans cette thèse, nous avons contribué au développement d'algorithmes biomimétiques inspirés du comportement collectif des fourmis. Cette contribution est une hybridation qui comprend deux étapes : dans un premier temps, nous initialisons avec un algorithme issu du domaine de la classification (l'algorithme K-Means) et dans deuxième temps en y introduisant un processus biomimétique de clustering dynamique, inspirés par le comportement collectif des fourmis dirigé par les traces d'odeurs coloniales. Cette approche d'hybridation améliore significativement les résultats de la classification obtenue. Nous avons également vu dans le chapitre 2 que les algorithmes de fourmis s'appliquaient aussi très bien à la résolution de problèmes d'optimisation. Un grand nombre de ces problèmes trouvent une solution par la modélisation de graphes. A cet effet, il serait intéressant d'envisager la résolution de problèmes d'optimisation par la construction des odeurs coloniales issue du système de reconnaissance chimique chez les fourmis sous forme de graphes.

Nous avons développé des travaux abordant un certain nombre de thématiques comme le partage d'une odeur coloniale commune par des fourmis formant un nid qui est à la base du système de reconnaissance et l'utilisation de phéromones pour la construction de structures, une nouvelle approche du partitionnement d'un ensemble de données sous la forme d'un graphe et la visualisation des résultats. Nous proposons finalement une nouvelle méthode de classification qui représente l'aspect incrémental des données à base de graphes qui nous permet également de traiter un volume de données important. Pour chacun de ces aspects, nous avons présenté nos conclusions.

En ce qui concerne la construction d'une odeur coloniale commune, nous

avons proposé un nouvel algorithme pour le domaine des fourmis artificielles. A notre connaissance, les méthodes à base de fourmis n'utilisaient pas jusque là ce modèle où les déplacements des fourmis se réalisent sur une structure de graphes dynamiques, et les approches dédiées à la classification restaient non hiérarchiques. Grâce à ce modèle, nous avons pu proposer une nouvelle méthode de construction de graphes dynamiques à partir de données. Les principales caractéristiques de ce nouveau modèle imitent les principes que l'on retrouve chez les vraies fourmis : une structure est construite sans contrôle central, et cette structure est formée par une étape initialisation faite par l'algorithme K-Means. Ces principes sont directement liés aux bons résultats rencontrés par notre méthode. Ce modèle est passionnant pour de nombreuses raisons, comme le soulignent les approches de robotique notamment. Son étude ne fait donc que commencer et nous dégagerons dans la section suivante de nombreuses perspectives. De point de vue de la classification non supervisée, nous avons donc introduit une nouvelle méthode dont les fourmis se déplacent dans un graphe dynamique. Globalement cette méthode se caractérise par le fait qu'elle nécessite d'information a priori (nombre de classes, partition initiale), et qu'elle est capable de traiter rapidement un grand nombre de données. Nous avons développé ainsi des modèles stochastiques, déterministes et réaliser des expériences sur des données classiques et réelles. D'après les comparaisons que nous avons pu faire, il apparaît que cette méthode est valide et opérationnelle. Nos résultats ont donc montré que notre méthode était compétitive avec la classification ascendante hiérarchique : les résultats ont une qualité comparable et sont obtenus en un temps nettement plus court. Cela nous a permis de développer une première version incrémentale de notre algorithme en comparant avec des algorithmes qui traitent les données en flux.

Par rapport aux méthodes utilisant le modèle des fourmis pour résoudre le problème de classification, nous utilisons d'une part un nouveau modèle comme mentionné précédemment, mais d'autre part d'autres atouts se dégagent : la nature du résultat (un graphe) les rend directement interprétables (par rapport aux travaux de [Lumer 1994b], [Azzag 2005] et [Lavergne 2008b]), nous pouvons traiter des données binaires (par rapport à [Monmarché 2000]), nous obtenons en un temps très court un graphe de données plutôt qu'un partitionnement (par rapport à [Labroche 2003]). Notre approche utilise une mesure de similarité qui dépend du type des données. Notre modèle est donc capable de classifier très rapidement des ensembles de données. Par rapport aux méthodes existantes, il nécessite de définir à l'avance un partitionnement en utilisant l'algorithme K-Means. Il est cependant certain que les caractéristiques de nos algorithmes de fourmis permettent de dégager les limites de

notre approche comme nous le verrons dans les perspectives.

Perspectives

En ce qui concerne les déplacements des fourmis dans une structure généralisée comme le graphe, plusieurs directions importantes peuvent être dégagées. La première consiste à développer une interface au lieu d'utiliser Tulip pour une représentation simple qui nous pourrions associer de manière visuelle une information plus riche afin de simuler les déplacements des fourmis et l'évolution du graphe après chaque étape de l'algorithme. Notre méthode réalise bien un partitionnement des données d'un ensemble. Cependant, nous aimerions également offrir à l'expert du domaine une classification automatique et présenter de manière visuelle le résultat sous la forme de clusters distincts dans une structure d'un graphe dynamique. En effet, nous désirons intégrer notre approche pour une visualisation de graphe de voisinage à l'aide d'une interface homme-machine.

L'adaptation des principes développés dans nos modèles à des données symboliques, comme peuvent l'être des données textuelles, trouvera certainement de nombreuses applications dans le domaine de la découverte de connaissances et la recherche d'informations sur internet, pour résoudre des problèmes industriels sera dans le futur à l'origine d'innovations les plus diverses et originales.
Ainsi vis-à-vis les paramètres dans nos méthodes, il faudrait envisager de prendre en compte un certain nombre de paramètres décrits mais ne sont pas bien établis comme le seuil dont lequel on décide de supprimer l'arête du graphe représentant une relation faible de proximité entre deux clusters voisins. Nous pourrions en ce sens améliorer les résultats en testant plusieurs valeurs de α et γ pour le paramètre de renforcement et celui de l'évaporation de phéromone.

Nous comptons appliquer notre algorithme à un volume de données important. En effet les résultats obtenus par notre version incrémentale CL-AntInc sont très encourageants et plusieurs améliorations peuvent y être apportées. Par exemple, en appliquant notre algorithme à des données en temps réel. Il serait donc évident d'éviter de stocker la totalité des informations nécessaires à réaliser la classification ou à manipuler les données en flux. Ceci permettrait une meilleure gestion en terme de stockage mémoire et un gain en temps d'exécution. En second lieu, l'algorithme CL-AntInc pourrait également être parallélisé pour accélérer et minimiser la complexité des traitements sur les

données volumineuses (la tendance de Big Data) en déportant les comportements des fourmis (déplacements, affectations) sur plusieurs machines et en les exécutant simultanément (en utilisant MapReduce par exemple).

Une étude comparative avec des algorithmes incrémentaux tel que DenStream ou D-Stream et CluStream (voir chapitre 5) permettrait une meilleure évaluation de notre méthode. Enfin, il serait également intéressant de tester d'autres types de données et d'autres mesures de distance. En effet nous pourrions améliorer la manière de simuler les données en flux. Il serait judicieux de penser à l'ordre dans lequel les données seront simulées.
Enfin, après toutes ces améliorations il serait possible de construire un système complet de gestion des données qui traite des différents types de données.

ANNEXE A

PUBLICATIONS SCIENTIFIQUES

Nous présentons ci-dessous les différentes publications scientifiques qui ont découlé de ce travail de thèse.

Revue internationale référencée (Thomson Reuters)

– N.Masmoudi, H. Azzag, M. Lebbah, C. Bertelle, M. Ben Jemaa : Clustering Numerical Data Using Artificial Ants, International Journal of Computer Science and Information Security, Vol. 14 No. 9, 2016.

Conférences internationales

– N.Masmoudi, H. Azzag, M. Lebbah, C. Bertelle, M. Ben Jemaa : CL-AntInc Algorithm for Clustering Binary Data Streams Using the Ants Behavior. The 20^{th} International Conference on Knowledge-Based and Intelligent Information and Engineering Systems Kes, Volume 96, pp 187196, York-United Kingdom, 2016.

– N.Masmoudi, H. Azzag, M. Lebbah, C. Bertelle, M. Ben Jemaa : Clustering of Binary Data Sets Using Artificial Ants Algorithm. The 22^{th} International Conference on Neural Information Processing ICONIP (1) : Chapter Neural Information Processing Volume 9489 of the series Lecture Notes in Computer Science pp 716-723, Turkey-Istanbul, 2015.

– N.Masmoudi, H. Azzag, M. Lebbah, C. Bertelle, M. Ben Jemaa : How to use ants for data stream clustering. The IEEE Congress on Evolutionary Computation CEC 2015 : pp 656-663, Japon-Sendai, 2015.

– N.Masmoudi, H. Azzag, M. Lebbah, C. Bertelle : Incremental clustering of data stream using real ants behavior. Proceedings of the 6^{th} World Congress on Nature and Biologically Inspired Computing NaBIC 2014 : pp 262-268, Portugal-Porto, 2014.

– N.Masmoudi, H. Azzag, M. Lebbah, C. Bertelle : Clustering using chemical and colonial odors of real ants. Proceedings of the 5^{th} World Congress on Nature and Biologically Inspired Computing NaBIC 2013 : pp 207-213, USA-Fargo, 2013.

En soumission

– N.Masmoudi, H. Azzag, M. Lebbah, C. Bertelle, M. Ben Jemaa : An Ant Based New Clustering Model for Graph Proximity Construction. International Journal of Bio-Inspired Computation, 2016.

Séminaires

– N.Masmoudi, présentation d'une étude bibliographique avancée, Différentes méthodes de classification de données et les fourmis artificielles, aux journées des systèmes complexes JSC-CST'2012 à la cité des sciences à Tunis, 2012.

BIBLIOGRAPHIE

[Abraham 2003] A. Abraham et V. Ramos. *Web Usage Mining Using Artificial Ant Colony Clustering and Linear Genetic Programming*. In The Congress on Evolutionary Computation, pages 1384–1391, Canberra, Australia, 08-12 December 2003. IEEE-Press. (Cité en page 61.)

[Admane 2006] L. Admane, K. Benatchba, M. Koudil, L. Siad et S. Mazi. *AntPart : an algorithm for the unsupervised classification problem using ants.* Laboratoire : Méthodes de Conception des Systèmes, de lInstitut National dInformatique, Oued-Smar, Alger, Algérie, 2006. (Cité en page 69.)

[Aggarwal 2003] Charu C. Aggarwal, Jiawei Han, Jianyong Wang et Philip S. Yu. *A framework for clustering evolving data streams*. In VLDB '2003 : Proceedings of the 29th international conference on Very large data bases, pages 81–92. VLDB Endowment, 2003. (Non cité.) :2003hl :2003hl :2003hl :2003hl

[Allen 1993] R.B. Allen, P. Obry et M. Littman. *An interface for navigating clustered document sets returned by queries*. In Proceedings of the conference on Organizational computing systems, pages 166–171. ACM Press, 1993. (Cité en page 49.)

[Anderson 2002] C. Anderson, G. Theraulaz et J. L. Deneubourg. *Self-assemblages in insect societies*. Insectes Sociaux, vol. 49, no. 2, pages 99–110, Mai 2002. (Cité en pages 32 et 65.)

[Angelov 2006] S. Angelov, S. Khanna et M. Visontai. *On the Complexity of Graph Self-assembly in Accretive Systems.* In Chengde Mao et Takashi Yokomori, editeurs, DNA, volume 4287 of *Lecture Notes in Computer Science*, pages 95–110. Springer, 2006. (Cité en page 84.)

[Ango. 1997] Ango., M. Revenu, A. Elmoataz et R. Clouard. *Les graphes de voisinage comme outil de mise en oeuvre de méthodes de segmentation hiérarchique d'images*. In 16e Colloque GRETSI, pages 399–402, Septembre 1997. (Cité en page 81.)

[Atlan 2006] H. Atlan. *L'organisation biologique et la théorie de l'information.* La librairie du XXI éme siècle. Seuil, 3 édition., 2006. (Cité en pages 46 et 48.)

[Auber 2002] D. Auber. *Outils de visualisation de larges structures de données.*. Thèse doctorat, Université Bordeaux I., 2002. (Cité en pages 72, 73, 113, 114 et 132.)

[Aufaure 2000] M.A. Aufaure, L. Yeh et K. Zeitouni. Le temps, l'espace et l'évolutif en sciences du traitement de l'information, volume 2, chapitre Fouille de données spatiales, pages 319–328. H. Prade and R. Jeansoulin and C. Garbay, septembre 2000. (Cité en page 81.)

[Azzag 2000] H. Azzag, F. Picarougne, C. Guinot et G. Venturini. *Un survol des algorithmes biomimétiques pour la classification.* Laboratoire d'Informatique de l'Université de Tour., 2000. (Cité en page 22.)

[Azzag 2004] H. Azzag, F. Picarougne, C. Guinot et G. Venturini. Revue des nouvelles technologies de l'information (rnti-c-1), volume Classification et fouille de données, chapitre Un survol des algorithmes biomimétiques pour la classification, pages 13–24. Cépaduès éditions, 2004. (Cité en page.) :2004dq22 :2004dq1 :2004dq22 :2004dq

[Azzag 2005] H. Azzag. *Classification hiérarchique par des fourmis artificielles : applications à la fouille de données et de textes pour le Web*. PhD thesis, Université François Rabelais de Tours, 2005. (Cité en pages 33 et 152.)

[Azzag 2007] H. Azzag, G. Venturini, A. Oliver et C. Guinot. *A hierarchical ant based clustering algorithm and its use in three real-world applications*. European Journal of Operational Research, vol. 179, no. 3, pages 906–922, 2007. (Cité en pages 4 et 66.)

[Bagnères 1998] A. Bagnères, G. Rivière et J. Clément. *Artificial neural network modeling of caste odor discrimination based on cuticular hydrocarbons in termites Chemoecology.* pages 201–209, 1998. (Cité en pages 20 et 34.)

[Beech 2004] J. Beech, P. Sommansson et M. Jansson. *Self-assembly project, theory and practice*. Rapport technique, Lunds University, Department of Physics, 2004. (Cité en page 84.)

[Bellot 2000] P. Bellot. *Méthodes de classification et de segmentation locales non supervisées pour la recherche documentaire*. PhD the-

sis, Université d'Avignon et des Pays de Vaucluse, 2000. (Cité en page 49.)

[Ben-Jacob 2000] E. Ben-Jacob, I. Cohen et H. Levine. *Cooperative self-organization of microorganisms.* Advances in Physics., pages 395–554, 2000. (Cité en page 16.)

[Beni 2005] G. Beni. *From swarm intelligence to swarm robotics.* Swarm Robotics., pages 1–9, 2005. (Cité en page 25.)

[Beshers 2001] S. N. Beshers et J. H. Fewell. *Models of division of labor in social insects.* Annual Review of Entomology, pages 413–440, 2001. (Cité en page 16.)

[Beyer 2002] H.-G. Beyer et H.-P. Schwefel. *Evolution strategies -a comprehensive introduction.* Natural Computing, 2002. (Cité en page 20.)

[Birattari 2007] M. Birattari, P. Pellegrini et M. Dorigo. *On the Invariance of Ant Colony Optimization*. IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION., pages 732–742, 2007. (Cité en page 59.)

[Blum 2004] C. Blum et M. Dorigo. *The Hyper-Cube Framework for Ant Colony Optimization.* IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICSPART B : CYBERNETICS, 34 (2)., pages 1161–1172, 2004. (Cité en page 59.)

[Blum 2008] C. Blum et D. Merkle. *Swarm Intelligence.* Natural Computing Series. Springer-Verlag., 2008. (Cité en page 15.)

[Boccara 2004] N. Boccara. *Modeling Complex Systems.* Springer, 2004. (Cité en page 21.)

[Bonabeau 1998] E. Bonabeau, G. Theraulaz et J.-L. Deneubourg. *The synchronization of recruitment-based activities in ants.* BioSystems, pages 195–211, 1998. (Cité en pages 16 et 84.)

[Bonabeau 1999a] E. Bonabeau, M. Dorigo et G. Theraulaz. *Swarm Intelligence.* From Natural to Artificial Systems., 1999. (Cité en pages 15, 16, 18, 23, 30 et 31.)

[Bonabeau 1999b] E. Bonabeau, M. Dorigo et G. Theraulaz. Swarm intelligence : From natural to artificial systems. Oxford University Press, New York, 1999. (Cité en page 57.)

[Bonner 1964] R.H. Bonner. *On some clustering techniques.* IBM Journal, 1964. (Cité en page 44.)

[Boulmerka 2009] A. Boulmerka. *Adaptation des métaheuristiques à l'ordonnancement hors-ligne des tâches temps réel à contraintes strictes en environnement monoprocesseur.* 2009. (Cité en pages 5 et 59.)

[Brandes 2003] U. Brandes, M. Gaertler et D. Wagner. *Experiments on Graph Clustering*. In Proceedings of the 11th Annual European Symposium on Algorithms (ESA'03), volume 2832, pages 568–579, 2003. (Cité en page 85.)

[Breed 1998] M.D. Breed. *Pheromone Communication in Social Insects.* Westview Press., 1998. (Cité en pages 28 et 34.)

[Brooks 1989] R. A. Brooks et A. M. Flynn. *Fast, cheap and out of control : a robot invasion of the solar system.* Journal of The British Interplanetary Society., page 478485, 1989. (Cité en pages 3 et 26.)

[Brossut 1996] R. Brossut. *Phéromones, la communication chimique chez les animaux.* CNRS editions, Belin., 1996. (Cité en page 28.)

[Buhl 2004] J. Buhl, J. Gautrais, R. V. Solé, P. Kuntz, S. Valverde, J.-L. Deneubourg et G. Theraulaz. *Efficiency and robustness in ant networks of galleries.* The European Physical Journal B - Condensed Matter and Complex Systems., pages 123–129, 2004. (Cité en page 16.)

[Buhl 2005] J. Buhl, J.-L. Deneubourg, A. Grimal et G. Theraulaz. *Self-organized digging activity in ant colonies.* Behavioral Ecology and Sociobiology., pages 9–17, 2005. (Cité en page 16.)

[Bullnheimer 1999] B. Bullnheimer et R. F. Hartl. *A New Rank Based Version of the Ant System.* A Computational Study. Central European Journal for Operations Research and Economics, 7(1)., pages 25–38, 1999. (Cité en page 59.)

[Burchard 1982] R. P. Burchard. *Trail following by gliding bacteria.* J. Bacteriol., pages 495–501, 1982. (Cité en page 29.)

[Burd 2006] M. Burd. *Ecological consequences of trafic organisation in ant societies.* Physica A : Statistical and Theoretical Physics., pages 124–131, 2006. (Cité en page 16.)

[Camazine 2001a] J.-L.and Franks N. R.and Sneyd J. Camazine S.and Deneubourg, G. Theraulaz et E. Bonabeau. *Self-Organization in Biological Systems*. Princeton studies in complexity. Princeton University Press, 2001. (Cité en pages 18, 20 et 31.)

[Camazine 2001b] S. Camazine, N.R. Franks, J. Sneyd, E. Bonabeau, J-L. Deneubourg et G. Theraula. Self-organization in biological systems. Princeton University Press, Princeton, NJ, USA, 2001. (Cité en pages 15, 16 et 18.)

[Camazine 2003] S. Camazine, J. Deneubourg, N. R. Franks, J. Sneyd, G. Theraulaz et E. Bonabeau. *Characteristics of Self-Organizing Systems*. chapitre 3. Princeton University Press., 2003. (Cité en page 18.)

[Candillier 2006] L. Candillier. *Contextualisation, visualisation et évaluation en apprentissage supervisé.* Thèse en université de Lille3., 2006. (Cité en page 43.)

[Cao 2006] F. Cao, M. Ester, W. Qian et A. Zhou. *Density-based clustering over an evolving data stream with noise*. page 328339, 2006. (Non cité.) :06 :06 :06 :06

[Carlin 1987] N.F. Carlin et B. Hölldobler. *The kin recognition system of carpenter ants (Camponotus. spp.)*. II. larger colonies, vol. 20, pages 209–217, 1987. (Cité en pages 34, 35 et 92.)

[Celeux 1989] G. Celeux et E. Diday. *Classification automatique de données environnement statistique et informatique.* Dunod, Informatique, 1989. (Cité en pages 3, 44, 46 et 47.)

[Charles 2004] C. Charles. *SearchXQ : une méthode daide 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 lEcole des Mines de Paris, spécialité "Informatique Temps Réel, Robotique et Automatique"., 2004. (Cité en page 40.)

[Cignoni 1998] P. Cignoni, C. Montani et R. Scopigno. *A fast divide and conquer delaunay triangulation algorithm.* Computer-Aided Design., 1998. (Cité en page 76.)

[Cit] *http ://citeseer.ist.psu.edu.* (Cité en page 85.)

[Colorni 1991] A. Colorni, M. Dorigo et V. Maniezzo. *Distributed Optimization by Ant Colonies*. In F.Varela et P.Bourgine, editeurs, Proceedings of the First European Conference on Artificial Life, pages 134–142, Paris, France, 1991. Elsevier Publishing. (Cité en pages 57 et 59.)

[Cordon 2000] O. Cordon, I. Viana, F. Herrera et L.A Moreno. *new ACO model integrating evolutionary computation concepts : The best-worst Ant System.* In Proc. ANTS 2000, M.Dorigo et al., Eds., IRIDIA, Université Libre de Bruxelles, Belgium., pages 22–29, 2000. (Cité en page 59.)

[Couzin 2003a] I. D. Couzin et N. R. Franks. *Self-organized lane formation and optimized traffic flow in army ants.* Proceedings of the Royal Society B Biological Sciences., pages 139–146, 2003. (Cité en page 16.)

[Couzin 2003b] I.D. Couzin et J. Krause. *Self-organization and collective behavior in vertebrates*. Advances in the Study of Behavior, pages 1–75, 2003. (Cité en page 16.)

[Cutting 1992] D.R. Cutting, D.R. Karger, J.O. Pedersen et J.W. Tukey. *Scatter/Gather : a cluster-based approach to browsing large document collections*. In Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval, pages 318–329. ACM Press, 1992. (Cité en page 49.)

[Dahabi 1998] A. Dahabi, P. Jaisson, A. Lenoir et A. Hafetz. *Comment les fourmis partagent leur odeurs.* La Recherche., pages 32–34, 1998. (Cité en page 34.)

[Danos 2005] V. Danos et F. Tarissan. *Self Assembling Graphs*. In José Mira et José R. Álvarez, editeurs, IWINAC (1), volume 3561 of *Lecture Notes in Computer Science*, pages 498–507. Springer, 2005. (Cité en page 84.)

[Dasgupta 1997] D. Dasgupta et N. Attoh-Okine. *Immune-based systems : A survey*. Proceedings of the IEEE International Conference on Systems, Man and Cybernetics., pages 369–374, 1997. (Cité en page 22.)

[Dasgupta 1999] D. Dasgupta. *Artificial Immune Systems and their applications.* Springer Verlag., 1999. (Cité en page 22.)

[De Castro 1999] L. N. De Castro et F. Von Zuben. *Artificial Immune Systems : Part I : Basic Theory and Applications.* Department of Computer Engineering and Industrial Automation, School of Electrical and Computer Engineering, State University of Campinas, Brazil., 1999. (Cité en page 22.)

[De Castro 2000a] L. N. De Castro et F. Von Zuben. *The clonal selection algorithm with engineering applications.* Proceedings of Genetic and Evolutionary Computation Conference. An Evolutionary Immune Network for Data Clustering., pages 36–37, 2000. (Cité en page 22.)

[De Castro 2000b] L. N. De Castro et F. Von Zuben. *An evolutionary immune network for data clustering.* In In Proceedings of the IEEE SBRN'00 (Brazilian Simposium on Artificial Neural Networks., pages 84–89, 2000. (Cité en page 22.)

[Deneubourg 1977] J.L. Deneubourg. *Application de l'ordre par fluctations à la description de certaines étapes de la construction du nid chez les termites.* Insectes Sociaux, pages 117–130, 1977. (Cité en page 30.)

[Deneubourg 1987] J.-L. Deneubourg, S. Goss, J. M. Pasteels, D. Fresneau et J.-P. Lachaud. *Self-organization mechanisms in ant societies (ii) : Learning in foraging and division of labor.* From individual cha-

racteristics to collective organisation : the example of social insects, pages 177–196, 1987. (Cité en page 16.)

[Deneubourg 1990] J.-L. Deneubourg, S. Goss, N.R. Franks, A. Sendova-Franks, C. Detrain et L. Chretien. *The dynamics of collective sorting : robot-like ant and ant-like robots.* In Proceedings of the First International Conference on Simulation of Adaptive Behavior., pages 356–365, 1990. (Cité en pages 29 et 31.)

[Deneubourg 1991] J.-L. Deneubourg, S. Goss, N. Franks, A. B. Sendova-Franks, C. Detrain et L. Chretien. *The dynamic of collective sorting robot-like ants and ant-like robots.* In J.A. Meyer and S.W. Wilson, editors, SAB'90 - 1st Conf. On Simulation of Adaptive Behavior : From Animals to Animats., pages 356–365, 1991. (Cité en pages 31, 61, 62 et 70.)

[Depickére 2004] S. Depickére, D. Fresneau et J.-L. Deneubourg. *Dynamics of aggregation in lasius niger (formicidae) : influence of polyethism.* Insectes Sociaux, pages 81–90, 2004. (Cité en page 31.)

[Dhillon 2005] I. Dhillon, Y. Guan et B. Kulis. *A fast kernel-based multilevel algorithm for graph clustering*. In KDD '05 : Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining, pages 629–634, New York, NY, USA, 2005. ACM. (Cité en pages 85 et 86.)

[Di Battista 1993a] G. Di Battista, P. Eades, R. Tamassia et I. G. Tollis. *Algorithms for drawing graphs : an annotated bibliography.* Preprint, Dept. Comput. Sci., Brown Univ., Providence, RI. Preliminary version available via anonymous ftp from wilma.cs.brown.edu, gdbiblio.tex.Z and gdbiblio.ps.Z in /pub/papers/compgeo., 1993. (Cité en page 79.)

[Di Battista 1993b] G. Di Battista, P. Eades, R. Tamassia et I. G. Tollis. *Graph Drawing : Algorithms for the Visualization of Graphs*. Prentice Hall PTR, Upper Saddle River, NJ, USA., 1993. (Cité en pages 72 et 80.)

[Di Battista 1994] G. Di Battista, P. Eades, R. Tamassia et I. G. Tollis. *Algorithms for drawing graphs : an annotated bibliography*. Comput. Geom. Theory Appl., 1994. (Cité en page 79.)

[Di Battista 1997] G. Di Battista. Graph drawing. (Proc. GD '97), volume 1353 de Lecture Notes Comput. Sci. Springer-Verlag, 1997. (Cité en page 79.)

[Di Caro 1997] G. Di Caro et M. Dorigo. *AntNet : A Mobile Agents Approach to Adaptive Routing.* Technical Report, IRIDIA, Université Libre de Bruxelles, Belgium., pages 97–12, 1997. (Cité en page 84.)

[Di Caro 998a] G. Di Caro et M. Dorigo. *AntNet : Distributed Stigmergetic Control for Communications Networks.* Journal of Artificial Intelligence Research., page 317365, 1998a. (Cité en page 84.)

[Di Caro 998b] G. Di Caro et M. Dorigo. *Mobile Agents for Adaptive Routing.* In 31st Hawaii International Conference on System, Big Island of Hawaii., pages 6–9, 1998b. (Cité en page 84.)

[Didimo 1997] W. Didimo et A. Leonforte. *GRID : An interactive tool for computing orthogonal drawings with the minimum number of bends.* In In Di Battista, G., éditeur : Graph Drawing (Proc. GD '97), Springer-Verlag., pages 309–315, 1997. (Cité en page 80.)

[Dorigo 1992] M. Dorigo. *Optimization, Learning and Natural Algorithms*. PhD thesis, Politecnico di Milano, Italy, 1992. In Italian. (Cité en page 57.)

[Dorigo 1996] M. Dorigo, V. Maniezzo et A. Colorni. *The Ant System : Optimization by a Colony of Cooperating Agents.* Systems, Man, and Cybernetics, Part B : Cybernetics, IEEE Transactions, Volume 26, Issue 1, pages 29–41, 1996. (Cité en pages 58 et 59.)

[Dorigo 1997] M. Dorigo et L. M. Gambardella. *Ant Colony System : A Cooperative Learning.* Approach to the Traveling Salesman Problem. IEEE Trans. Evol. Comp. tome 1., pages 53–66, 1997. (Cité en page 59.)

[Dorigo 2004] M. Dorigo et T. Stutzle. *Ant Colony Optimization.* The MIT Press., pages 36–37, 2004. (Cité en pages 23 et 57.)

[Dorigo 2005] M. Dorigo, E. Tuci, R. Grofi, V. Trianni, T. H. Labella, T. Halva, S. Nouyan, C. Ampatzis, J.L. Deneubourg, G. Baldassarre, S. Francesco Nolfi, F. Mondada, D. Floreano et L. M. Gambardella. *The swarm-bot project.* In Kunstliche Intelligenz, Springer Verlag., pages 31–44, 2005. (Cité en page 25.)

[Dorigo 2006] M. Dorigo, M. Birattari et T. Stützle. *Ant colony optimization- artificial ants as a computational intelligence technique.* IEEE Computational Intelligence Magazine., pages 28–39, 2006. (Cité en pages 32 et 57.)

[Dréo 2003] J. Dréo, A. Pétrowski, P. Siarry et E. Taillard. *Métaheuristiques pour l'optimisation difficile.* Eyrolles., 2003. (Cité en page 57.)

[Dréo 2004] J. Dréo et S Siarry. *Adaptation de la méthode des colonies de fourmis pour loptimisation en variables continues.* Application en génie biomédical, 2004. (Cité en pages 3 et 60.)

[Drogoul 1999] A. Drogoul et D. Fresneau. *Métaphore de fourragement et modèle d'exploitation collective de l'espace pour des colonies de robots autonomes mobiles*. Actes des JFIADSMA'98, 1999. (Non cité.) :1999xi :1999xi :1999xi :1999xi

[Dussutour 2004] A. Dussutour, V. Fourcassié, D. Helbing et J.-L. Deneubourg. *Efficiency and robustness in ant networks of galleries.* The European Physical Journal B - Condensed Matter and Complex Systems., pages 123–129, 2004. (Cité en page 16.)

[Dutot 2005] A. Dutot. *Distribution dynamique adaptative à l'aide de mécanismes d'intellignece collective, Détection d'organisations par des techniques de collaboration et de compétition.* Thèse doctorat., 2005. (Cité en page 67.)

[Eades 1989] P. Eades et R. Tamassia. *Algorithms for drawing graphs : An annotated bibliography.* Report CS-09-89, Department of Computer Science, Brown University, Providence, RI., 1989. (Cité en page 79.)

[Eades 1997] P. Eades, R. F. Cohen et M. L. Huang. *Online animated graph drawing for web navigation.* In Di Battista, G., éditeur : Graph Drawing (Proc. GD '97), volume 1353 de Lecture Notes Comput. Sci., pages 330–335, 1997. (Cité en page 80.)

[Edelsbrunner 1987] Herbert Edelsbrunner. Algorithms in combinatorial geometry. Springer-Verlag New York, Inc., New York, NY, USA, 1987. (Cité en page 75.)

[Eisen 1998] M.B Eisen, P.T. Spellman et Botstein D. *cluster analysis and display of genomewide expression patterns.* proceedings of the national academy of sciences(PNAS), Etat-unis, 1998. (Cité en page 49.)

[Ester 1997] M. Ester, H-P. Kriegel et J. Sander. *Spatial Data Mining : A Database Approach*. In Proceedings of the 5th Int. Symposium on Large Spatial Databases (SSD '97), pages 47–66, 1997. (Cité en page 81.)

[Everitt 1993] B.S. Everitt. Cluster analysis. Edward Arnold and Halsted Press, 1993. (Cité en pages 46 et 48.)

[Fisher 1991] D. H. Fisher. *Knowledge Acquisition via Incremental Conceptual Clustering*. Machine Learning, vol. 2, no. 2, pages 139–172, 1991. (Cité en page 50.)

[Flake 2000] G. W. Flake, S. Lawrence et C. L. Giles. *Efficient identification of web communities.* In KDD '00 : Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining., pages 150–160, 2000. (Cité en page 85.)

[Flake 2003] G. W. Flake, R. E. Tarjan et K. Tsioutsiouliklis. *Graph clustering and minimum cut trees.* Internet Mathematics., 2003. (Cité en page 86.)

[Forrest 1991] S. Forrest. *Emergent Computation : Self-organizing, Collective and Cooperative Phenomena in Natural and Artificial Computing Networks.* Special Issues of Physica D. MIT Press., 1991. (Cité en page 20.)

[Fortune 1997] Steven Fortune. Voronoi diagrams and delaunay triangulation. Handbook of Discrete and Computational Geometry, CRC. J. E. Goodman and J. O'Rourke, 1997. (Cité en page 75.)

[Franks 1992] N. R. Franks et A. Sendova-Franks. *Brood sorting by ants : Distributing the workload over the work-surface.* Behavioral Ecology and Sociobiology, pages 109–123, 1992. (Cité en pages 31, 59 et 61.)

[Fresneau 1994] D. Fresneau. *Biologie et comportement social d'une fourmi ponérine néotropicale (Pachycondyla apicalis).* PhD thesis, Thèse d'état-Université Paris Nord, 1994. (Non cité.) :1994jb :1994jb :1994jb :1994jb

[Frey] B. J. Frey et D. Dueck. *Clustering by passing messages between data points.* CNRS editions, Belin. (Cité en pages 4 et 82.)

[Gabriel 1969] K. R. Gabriel et R. R. Sokal. *A new statistical approach to geographic variation analysis*. In Systematic zoology, chapitre 18, pages 259–278. 1969. (Cité en page 77.)

[Galef 1996] J. B. G. Galef et L. L. Buckley. *Use of foraging trails by Norway rats.* pages 765–771, 1996. (Cité en page 29.)

[Gambardella 1995] L.M. Gambardella et M. Dorigo. *Ant-Q : A Reinforcement Learning Approach to the Travelling Salesman Problem*. In A. Prieditis et S. Russell, editeurs, Proceedings of the Twelfth International Conference on Machine Learning, pages 252–260. Morgan Kaufmann, San Mateo, California, 1995. (Cité en page 59.)

[Garnier 2008] S. Garnier. *Décisions collectives dans des systèmes d'intelligence en essaim*. Thèse doctorat à l'Université Toulouse III - Paul Sabatier, 2008. (Cité en pages 3 et 17.)

[Garzon 1993] Max H. Garzon. *Cayley automata*. Theor. Comput. Sci., vol. 108, no. 1, pages 83–102, 1993. (Cité en page 84.)

[Garzon 1999] M.H. Garzon, R.J. Deaton et K. Barnes. *On Self-Assembling Graphs in vitro*. In Wolfgang Banzhaf, Jason Daida, Agoston E. Eiben, Max H. Garzon, Vasant Honavar, Mark Jakiela et Robert E. Smith, editeurs, Proceedings of the Genetic and Evolutionary Computation Conference, volume 2, pages 1805–1809, Orlando, Florida, USA, 13-17 Juillet 1999. Morgan Kaufmann. (Cité en page 84.)

[Gordon 1996] D. M. Gordon. *The organization of work in social insect colonies.* Nature, pages 121–124, 1996. (Cité en pages 16 et 43.)

[Grassé 1939] P.P. Grassé. *La reconstruction du nid et le travail collectif chez les termites supérieurs Y.Psychol.* pages 370–396, 1939. (Cité en pages 28 et 30.)

[Grassé 1959] P.-P. Grassé. *La reconstruction du nid et les coordinations interindividuelles chez bellicosi- termes natalensis et cubitermes sp.* la theorie de la stigmergie : essai d'interpretation des termites constructeurs. Insectes sociaux, IV(1)., pages 41–83, 1959. (Cité en pages 18 et 29.)

[Guelzim 2002] N. Guelzim, S. Bottani, P. Bourgine et F. Kepes. *Topological and casual structure of the yeast transcriptional regulatory network.* Nature genetics, pages 60–63, 2002. (Cité en page 81.)

[Gueron 1993] L. S. Gueron S. A. *Self-organization of front patterns in large wildebeest herds*. Journal of theoretical biology, pages 541–552, 1993. (Cité en page 16.)

[Guha 1998] S. Guha, R. Rastogi et K. Shim. *CURE : an efficient clustering algorithm for large databases*. In Laura M. Haas et Ashutosh Tiwary, editeurs, Proceedings ACM SIGMOD International Conference on Management of Data, pages 73–84, Seattle, Washington, USA, 6 1998. ACM Press. (Non cité.) :1998df :1998df :1998df :1998df

[Hacid 2005] H. Hacid et D-A. Zighed. *An Effective Method for Locally Neighborhood Graphs Updating.* In DEXA 2005, pages 930–939, 2005. (Cité en page 77.)

[Hacid 2006] H. Hacid et D-A. Zighed. *Graphes de Proximité pour l'Indexation et l'Interrogation d'Images par le Contenu*. In 6èmes Journées Francophones Extraction et Gestion des Connaissances (EGC 06), Lille, volume E-6 of *Revue des Nouvelles Technologies de l'Information*, pages 11–22, Toulouse, Janvier 2006. Cépaduès. (Cité en pages 79 et 81.)

[Hacid 2007] H. Hacid et T. Yoshida. *Incremental Neighborhood Graphs Construction for Multidimensional Databases Indexing*. In the Canadian Artificial Intelligence Conference, CanAI 2007, LNAI 4509,

pages 405–416, Montreal, Quebec, Canada, May 2007. (Cité en page 76.)

[Haken 2008] H. Haken. *Self-organization. Scholarpedia.* 2008. (Cité en page 17.)

[Halloy 2007] J. Halloy, G. Sempo, G. Caprari, C. Rivault, M. Asadpour, F. Tache, I. Said, V. Durier, S. Canonge, J. M. Ame, C. Detrain, N. Correll, A. Martinoli, F. Mondada, R. Siegwart et J. L. Deneubourg. *Social Integration of Robots into Groups of Cockroaches to Control Self-Organized Choices.* Science., page 11551158, 2007. (Cité en pages 3 et 26.)

[Hartigan 1975] J.A Hartigan. *clustering algorithms.* John wiley and Sons, New York., 1975. (Cité en page 47.)

[Hartigan 1979] J. A. Hartigan et M. A. Wong. *A k-means clustering algorithm*. JSTOR : Applied Statistics, vol. 28, no. 1, pages 100–108, 1979. (Non cité.) :1979ct :1979ct :1979ct :1979ct

[Hartuv 1999] E. Hartuv et R. Shamir. *A clustering algorithm based on graph connectivity.* Information Processing Letters., 1999. (Cité en page 72.)

[Helbing 2001] D. Helbing, P. Molnàr, I. J. Farkas et K. Bolay. *Self-organizing pedestrian movement*. Environment and Planning B : Planning and Design, pages 361–383, 2001. (Cité en page 16.)

[Heusse 1998] M. Heusse, D. Snyers, S. Guerin et P. Kuntz. *Adaptive Agent Driven Routing and Load Balancing in Communication Networks.* Technical Report RR-98001-IASC, ENST de Bretagne., 1998. (Cité en page 84.)

[Holland 1992] J. H. Holland. *Adaptation in Natural and Artificial Systems : An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence.* The MIT Press, 2 édition., 1992. (Cité en pages 3, 20 et 26.)

[Holldobler 1990] B. Holldobler et E. Wilson. *The Ants.* Springer Verlag, Berlin, Germany., 1990. (Cité en pages 29 et 56.)

[Hundack 1997] C. Hundack, P. Mutzel, I. Pouchkarev et S. Thome. *ArchE : A graph drawing system for archaeology.* In In Di Battista, G., éditeur : Graph Drawing (Proc. GD '97), Springer-Verlag., pages 297–302, 1997. (Cité en page 80.)

[Jain 1988] A.K. Jain et R.C. Dubes. Algorithms for clustering data. Prentice Hall Advanced Reference Series, 1988. (Cité en pages 3, 41, 42, 44, 46 et 127.)

[Johnson 1967] S.C Johnson. *Hierarchical Clustering Schemes.* Psychometrika., 1967. (Cité en page 47.)

[Jolion 2003] J-M. Jolion. *Similarité entre images : primitives, graphes et vision humaine*, 23 novembre 2003. (Cité en page 81.)

[Jouve] B. Jouve, P. Kuntz et F. Velin. *Extraction de structures macroscopiques dans des grands graphes par une approche spectrale.* In Hérin, D. et Zighed, D. A., éditeurs : EGC, volume 1 de Extraction des Connaissances et Apprentissage. Hermes Science Publications., pages 173–184. (Cité en page 85.)

[Kannan 2000] R. Kannan, S. Vempala et A. Veta. *On clusterings-good, bad and spectral*. In FOCS '00 : Proceedings of the 41st Annual Symposium on Foundations of Computer Science, page 367, Washington, DC, USA, 2000. IEEE Computer Society. (Cité en page 85.)

[Karlson 1958] P. Karlson et M. Luscher. *Pheromones : a new term for a class of biologically active substances.* Nature., 1958. (Cité en page 29.)

[Karypis 998a] G. Karypis et V. Kumar. *A fast and high quality multilevel scheme for partitioning irregular graphs.* SIAM J. Sci. Comput., 1998a. (Cité en page 85.)

[Kaufman L. 1990] Rousseeuw P. J. Kaufman L. *Finding Groups in Data.* John Willey et Sons., 1990. (Cité en page 43.)

[Kaufmann 2001] M. Kaufmann et D. Wagner. *Drawing Graphs, Methods and Models.* Lecture Notes in Computer Science. Springer., 2001. (Cité en page 80.)

[Kerr 2000] M. K. Kerr et G. A. Churchill. *Bootstrapping cluster analysis : Assessing the reliability of conclusions from microarray experiments.* PNAS., 2000. (Cité en page 81.)

[Ketterlin 1994] A. Ketterlin et J. J. Korczak. *Concept formation in complex domains.* ECML, Vol. 784 Springer-Verlags LNCS, Berlin., 1994. (Cité en page 53.)

[Khedam 2008] R. Khedam. *Contribution au développement de méthodologies de fusion/classification contextuelle d'images satellitaires multisources.* Thèse doctorat à l'université des Sciences et de la Technologie Houari BOUMEDIENE., 2008. (Cité en pages 3 et 32.)

[Klavins 2002] E. Klavins. *Automatic Synthesis of Controllers for Distributed Assembly and Formation Forming.* In ICRA, pages 3296–3302. IEEE, 2002. (Cité en page 84.)

[Klavins 2004] E. Klavins, R. Ghrist et D Lipsky. *Graph Grammars for Self Assembling Robotic Systems.* In ICRA, pages 5293–5300. IEEE, 2004. (Cité en page 84.)

[Kohonen 2001] T. Kohonen. Self-organizing maps. Springer, Berlin, 2001. (Cité en page 54.)

[Kranen 2011] P. Kranen, I. Assent, C. Baldauf et T. Seidl. *The clustree : indexing micro-clusters for anytime stream mining*. Knowledge and Information Systems, vol. 29, page 249272, 2011. (Cité en page 54.)

[Krasnogor 2005] N. Krasnogor et M. Gheorghe. *Systems Self-Assembly*. International Workshop, The Grand Challenge in Non-Classical Computation, 18-19th April 2005, the University of York, United-Kingdom, April 2005. (Cité en pages 84 et 85.)

[Kuntz 1997] P. Kuntz, P. Layzell et D. Snyers. *A Colony of Ant-like Agents for Partitioning in VLSI Technology*. In P. Husbands et I. Harvey, editeurs, Proceedings of the Fourth European Conference on Artificial Life, pages 417–424, 1997. (Cité en pages 61 et 62.)

[Kuntz 1998] P. Kuntz, D. Snyers et P.J. Layzell. *A Stochastic Heuristic for Visualising Graph Clusters in a Bi-Dimensional Space Prior to Partitioning.* J. Heuristics, vol. 5, no. 3, pages 327–351, 1998. (Cité en page 32.)

[Kuntz 2000] P. Kuntz et F. Henaux. *Numerical comparison of two spectral decomposition for vertex clustering.* In Verlag, S., éditeur : Data Analysis, Classification and Related Methods., 2000. (Cité en page 85.)

[Labroche 2003] N. Labroche. *Modélisation du système de reconnaissance chimique des fourmis pour le problème de la classification non-supervisée : application à la mesure d'audience sur Internet*. Thèse de doctorat, Laboratoire d'Informatique, Université de Tours, décembre 2003. (Cité en pages 35, 40, 62, 70, 92 et 152.)

[Lambrinos 2000] D. Lambrinos, R. Moller, T. Labhart, R. Pfeifer et R. Wehner. *A mobile robot employing insect strategies for navigation.* Robotics and Autonomous Systems., page 3964, 2000. (Cité en pages 3 et 26.)

[Lavergne 2006] J. Lavergne. *Méthode de classification incrémentale par nuage d'insectes*. In Xe Forum de l'Ecole Doctorale Sante, Sciences, Technologies. Université François Rabelais de Tours, 23-24 Mai 2006. (Cité en page 84.)

[Lavergne 2007] J. Lavergne, H. Azzag, C. Guinot et G. Venturini. *Incremental Construction of Neighborhood Graphs Using the Ants Self-Assembly Behavior*. 19th IEEE International Conference on Tools

with Artificial Intelligence (ICTAI 2007), vol. 1, pages 399–406, 2007. (Cité en page 85.)

[Lavergne 2008a] J. Lavergne. *Algorithmes de fourmis artificielles pour la construction incrémentale et la visualisation interactive de grands graphes de voisinage*. PhD thesis, Université François-Rabelais de Tours, décembre 2008. (Cité en page 81.)

[Lavergne 2008b] J. Lavergne. *Algorithmes de fourmis artificielles pour la construction incrémentale et la visualisation interactive de grands graphes de voisinage*. PhD thesis, Université François Rabelais de Tours, 2008. (Cité en pages 33 et 152.)

[Lin 1997] C-R. Lin et M. Gerla. *Adaptive clustering for mobile wireless networks*. IEEE Journal on Selected Areas in Communications, vol. 15, pages 1265–1275, 1997. (Cité en page 86.)

[Lioni 2000] A. Lioni. *Auto-assemblage et transport collectif chez oecophylla*. PhD thesis, Université libre de bruxelles, Université Paul Sabatier, 2000. (Cité en page 65.)

[Lioni 2001] A. Lioni, C. Sauwens, G. Theraulaz et J.-L. Deneubourg. *The dynamics of chain formation in Oecophylla longinoda*. Journal of Insect Behavior, vol. 14, pages 679–696, 2001. (Cité en page 66.)

[Lumer 1994a] E. D. Lumer et B. Faieta. *Diversity and Adaptation in Populations of Clustering Ants*. In D. Cliff, P. Husbands, J.A. Meyer et Stewart W., editeurs, Proceedings of the Third International Conference on Simulation of Adaptive Behavior, pages 501–508. MIT Press, Cambridge, Massachusetts, 1994. (Cité en page 62.)

[Lumer 1994b] E.D. Lumer et B. Faieta. *Diversity and adaptation in populations of clustering ants.* In D. Cliff, P. Husbands, J.A. Meyer, and S.W. Wilson, editors, SAB'94 - 3rd International Conference on Simulation of Adaptive Behavior : From Animals to Animats., pages 501–508, 1994. (Cité en pages 31, 32, 61 et 152.)

[MacQueen 1967] J.B. MacQueen. *Some methods of classification and analysis of multivariate observations*. In Proceedings of 5th Berkley Symposium on Mathematical Statistics and Probability, pages 281–297, 1967. (Cité en pages 44, 90 et 96.)

[Martí 2003] R. Martí et M. Laguna. *Heuristics and meta-heuristics for 2-layer straight line crossing minimization.* Discrete Appl. Math., pages 665–678, 2003. (Cité en page 80.)

[Martinetz 1991] T. Martinetz et K. J. Schulten. *A neural-gas network learns topologies.* In T. Kohonen, K. Mäkisara, O. Simula, and J. Kan-

gas, editors, Artificial Neural Networks. Elsevier Science Publishers, Amsterdam,, page 397402, 1991. (Cité en page 83.)

[Martinetz 1993] T. Martinetz. *Competitive Hebbian Learning rule forms perfectly topology preserving maps.* In S. Gielen and B. Kappen, editors, Proceedings of the International Conference on Artificial Neural Networks (ICANN-93)., page 427434, 1993. (Cité en page 83.)

[Matula 1977] D. W. Matula. *Graph Theoretic Techniques for Cluster Analysis Algorithms.* In J. V. Ryzin, editor, Classification and Clustering. Academic Press, New York., 1977. (Cité en page 72.)

[Maziz 2004] S. Maziz et L. Siad. *AntMining*. PhD thesis, OuedSmar, Alger, Algérie, 2004. (Cité en pages 35, 65, 69 et 70.)

[Melhuish 2001] C. Melhuish, M. Wilson et A. Sendova-Franks. *Advances in artificial life.* In J. Kelemen and P. Sosík, editors, 6th European Conference ECAL 2001, pages 10-14, Prague, Czech Republic, Springer-Verlag GmbH., 2001. (Cité en page 32.)

[Michelsen 1989] D. A. Michelsen. *http ://robobees.seas.harvard.edu.* 1989. (Cité en pages 3 et 24.)

[Millonas 1993] M. M. Millonas. *Swarms, phase transitions, and collective intelligence.* In eprint arXiv :adap-org/9306002., 1993. (Cité en page 27.)

[Monmarché 2000] N. Monmarché. *Algorithmes de fourmis artificielles : applications à la classification et à l'optimisation.* Thèse doctorat., 2000. (Cité en pages 3, 34, 57, 60, 62, 63, 64 et 152.)

[Monmarché 2000a] N. Monmarché, G. Venturini et M. Slimane. *On how Pachycondyla apicalis ants suggest a new search algorithm*. Future Generation Computer Systems, vol. 16, no. 8, pages 937–946, 2000. (Cité en pages 59, 61 et 63.)

[Monmarché 2000b] Nicolas Monmarché. *Algorithme de fourmis artificielles : applications à la classification et à l'optimisation*. These de doctorat, Université de Tours, 2000. (Cité en page 31.)

[Munzner 1998] T. Munzner. *Drawing large graphs with h3viewer and site manager.* In Sue Whitesides, editor, 6th Symp. Graph Drawing, Lecture Notes in Computer Science., 1998. (Cité en page 81.)

[Nasaroui 2002] O. Nasaroui, D. Dasgupta et F. Gonzalez. *The fuzzy artificial immune system : Motivations, basic concepts, and application to clustering and web profiling.* In Proceedings of the IEEE International Conference on Fuzzy Systems at WCCI., pages 711–716, 2002. (Cité en page 22.)

[Nicolis 1977] G. Nicolis et I. Prigogine. *Self-organization in Non-equilibrium Systems.* NewYork, 1977. (Cité en page 20.)

[Okabe 1992] A. Okabe, B. Boots et K. Sugihara. Spatial tessellations : concepts and applications of voronoi diagrams. John Wiley & Sons, Inc., New York, NY, USA, 1992. (Cité en page 76.)

[Passera 1958] L. Passera. *L'organisation sociale des fourmis.* Nature., 1958. (Cité en page 29.)

[Patrignani 1997] M. Patrignani et F. Vargiu. *3DCube : A tool for three dimensional graph drawing.* In In Di Battista, G., éditeur : Graph Drawing (Proc. GD '97), pages 284–290, 1997. (Cité en page 80.)

[Picarougne 2004] F. Picarougne, H. Azzag, G. Venturini et C. Guinot. *On Data Clustering with a Flock of Artificial Agents*. In Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence (ICTAI'04), pages 777–778, Boca Raton, Florida, USA, 2004. IEEE Computer Society. (Cité en page 34.)

[Poincot 1998] P. Poincot, S. Lesteven et F. Murtach. *A spatial user interface to the astromical Literature.* Astronomy and astophysics supplement series., 1998. (Cité en page 55.)

[Preparata 1985] F. P. Preparata et M. I. Shamos. Computational geometry : Introduction (monographs in computer science). Springer-Verlag, 1985. (Cité en page 76.)

[Prigogine 1971] I. Prigogine et P. Glandsdorf. *Thermodynamic Theory and Structure.* Stability and Fluctuations, Wiley and Sons, New York., 1971. (Cité en page 20.)

[Ralambondrainy 1995] H. Ralambondrainy. *A conceptual version of the k-means algorithm. In Pattern Recognition Lett.* vol. 16, page 11471157, 1995. (Cité en page 42.)

[Ramos 2002] V. Ramos et J.J. Merelo. *Self-organized stigmergic document maps :Environment as mechanism for context learning*. In Proceedings of the First Spanish BIBLIOGRAPHIE 116 Conference on Evolutionary and Bio-Inspired Algorithms (AEB 2002), Centro Univ. de Mérida, Mérida, Spain,E. ALBA, F. HERRERA, J.J. MERELO et al, pages 6–8, 2002. (Cité en page 61.)

[Rechenberg 1973] I. Rechenberg. *Evolutionsstrategie : Optimierung technischer Systeme nach Prinzipien der biologischen Evolution.* rommann-Holzboog Verlag., 1973. (Cité en page 20.)

[Reynolds. Flocks 1987] C. W. Reynolds. Flocks. *A distributed behavioral model.* Computer Graphics (SIGGRAPH '87 Conference Proceedings)., pages 25–34, 1987. (Cité en page 24.)

[Robinson] A. Robinson. *Visualisation of microarray gene expression data.* http ://industry.ebi.ac.uk/alan/. (Cité en page 81.)

[Rodolphe 2009] C. Rodolphe. *L'intelligence en essaim sous l'angle des systèmes complexes : étude d'un système multi-agent réactif à base d'itérations logistiques couplées.* Thèse doctorat à l'université Nancy 2., décembre 2009. (Cité en page 21.)

[Saha 2006] B. Saha et P. Mitra. *Fast Incremental Minimum-Cut Based Algorithm for Graph Clustering*. In IEEE International Workshop on Mining Evolving and Streaming Data, Hong Kong, December, 18-22 2006. The 2006 IEEE International Conference on Data Mining. (Cité en pages 80 et 86.)

[Sahin 2008] E. Sahin, S. Girgin, L. Bayindir et A. E. Turgut. *Swarm robotics.* Swarm Intelligence, Natural Computing Series. Springer-Verlag., pages 87–100, 2008. (Cité en pages 23 et 25.)

[Sauwens 2000] C. Sauwens. *Étude de la dynamique dauto-assemblage chez plusieurs espèces de fourmis.* Thèse de doctorat, Université libre de bruxelles., 2000. (Cité en page 65.)

[Schoonderwoerd 1997] R. Schoonderwoerd, O. Holland, J. Bruten et L. Rothkrantz. *Ant-based Load Balancing in Telecommunications Networks.* Adaptive Behavior., page 169207, 1997. (Cité en page 84.)

[Sigel 2000] E. Sigel, B. Denby et S. Le Hegarat-Mascle. *Application of Ant Colony Optimization to Adaptive Routing in LEO Telecommunications Satellite Network.* Submitted to IEEE Transactions on Networking., 2000. (Cité en page 84.)

[STÜTZLE 1999] DORIGO M. STÜTZLE T. *ACO Algorithms for the Traveling Salesman Problem.* Evolutionary Algorithms in Engineering and Computer Science : Recent Advances in Genetic Algorithms, Evolution Strategies, Evolutionary Programming, Genetic Programming and Industrial Application.., 1999. (Cité en page 56.)

[Stützle 1997] T. Stützle et H. Hoos. $\mathcal{MAX} - \mathcal{MIN}$ *Ant System and Local Search for the Traveling Salesman Problem.* In Proceedings of the fourth International Conference on Evolutionary Computation, pages 308–313. IEEE Press, 1997. (Cité en page 59.)

[Sumpter 2006] D. J. T. Sumpter. *The principles of collective animal behaviour*. Philosophical Transactions of the Royal Society of London B, pages 5–22, 2006. (Cité en page 16.)

[Taillard 1998] E.D. Taillard, L. Gambardella et J-Y. Gendreau M. Potvin. *Adaptive Memory Programming : A Unied View of Metaheuristics.*

EURO XVI Conference Tutorial and Research Reviews booklet, Brussels, 1998. (Cité en page 63.)

[Tarissan 2006] F. Tarissan. *Étude d'un formalisme concurrent pour les phénomènes d'auto-organisation et la biologie moléculaire*. PhD thesis, Université Paris 7 - Denis Diderot - UFR d'Informatique, 13 décembre 2006. (Cité en page 84.)

[Theraulaz 1995] G. Theraulaz et E. Bonabeau. *Coordination in distributed building*. Science., pages 686–688, 1995. (Cité en pages 18 et 30.)

[Theraulaz 1997] G. Theraulaz et F. Spitz. *Auto-organisation et comportement*. Hermès, Paris, page 320, 1997. (Cité en page 15.)

[Theraulaz 2001] G. Theraulaz, E. Bonabeau, C. Sauwens, J.-L. Deneubourg, A. Lioni, F. Libert, L. Passera et R.-V. Solé. *Model of droplet formation and dynamics in the Argentine ant (Linepithema humile Mayr)*. Bulletin of Mathematical Biology, 2001. (Cité en page 66.)

[Theraulaz 2003] G. Theraulaz, J. Gautrais, S. Camazine et J.-L. Deneubourg. *The formation of spatial patterns in social insects : from simple behaviours to complex structures*. Philosophical Transaction of the Royal Society A, pages 1263–1282, 2003. (Cité en page 16.)

[Théraulaz 1997] G. Théraulaz et E. Bonabeau. *Auto-organisation et comportement*. chapter Auto-Organisation et comportement collectifs : la modélisation des sociétés d'insectes., pages 91–140, 1997. (Cité en page 18.)

[Timmis 2000] J. Timmis et M. Neal. *An Artificial Immune System for Data Analysis*. BioSystems., pages 143–150, 2000. (Cité en page 22.)

[Timmis 2001] J. Timmis et T. Knight. *Artificial immune systems : Using the immune system as inspiration for data mining*. In Hussein A. Abbass, Ruhul A. Sarker, and Charles S. Newton, editors, Data Mining : A Heuristic Approach, chapter XI., pages 209–230, 2001. (Cité en page 22.)

[Timmis 2002] J. Timmis et T. Knight. *On data clustering with artificial ants*. Proceedings of the 4th International Conference on Recent Advances in Soft Computing., pages 266–271, 2002. (Cité en page 22.)

[Toussaint 1980] G. T. Toussaint. *The relative neighborhood graphs in a finite planar set*. In Pattern recognition, chapitre 12, pages 261–268. 1980. (Cité en page 78.)

[Toussaint 1991] G. T. Toussaint. *Some unsolved problems on proximity graphs*. In D. W. Dearholt et F. Harrary, editeurs, Memoranda in computer and cognitive science MCCS-91-224, Computing research

laboratory, New Mexico state University Las Cruces, 1991. (Cité en page 73.)

[Tran Dac 2004] H. Tran Dac. *Sectorisation contrainte de l'espace aérien*. PhD thesis, Université de Technologie de Compiègne, mai 2004. (Cité en pages 76 et 81.)

[Tschinkel 2004] W. R. Tschinkel. *The nest architecture of the forida harvester ant, pogonomyrmex badius.* Journal of Insect Science, pages 4–12, 2004. (Cité en page 16.)

[Tu 2009] L. Tu et Y. Chen. *Stream data clustering based on grid density and attraction*. ACM Transactions on Knowledge Discovery Data, no. 3, page 127, 2009. (Cité en pages 53 et 136.)

[Vittori 2006] K. Vittori, G. Talbot, J. Gautrais, V. Fourcassié, A. F. R. Araujo et G. Theraulaz. *Path effciency of ant foraging trails in an artificial network*. Journal of Theoretical Biology., pages 507–515, 2006. (Cité en page 16.)

[Winfree 1998] E. Winfree. *Algorithmic self-assembly of dna*. PhD thesis, Pasadena, CA, USA, 1998. Adviser-John J. Hopfield. (Cité en page 84.)

[Wong 2007a] W. Wong, W. Liu et M. Bennamoun. *Tree-Traversing Ant Algorithm for term clustering based on featureless similariti.* Springer Science+Business Media., 2007. (Cité en page 49.)

[Wong 2007b] W. Wong, W. Liu et M. Bennamoun. *Tree-Traversing Ant Algorithm for term clustering based on featureless similaritie.* Springer Science+Business Media, LLC., 2007. (Cité en pages 4, 66 et 68.)

[Wu 1993] Z. Wu et R. Leahy. *An approximation method of evaluating the joint likelihood for first-order GMRFs.* IEEE Transactions on Image Processing., 1993. (Cité en page 72.)

[Yee 2001] K.-P. Yee, D. Fisher, R. Dhamija et M. A. Hearst. *Animated exploration of dynamic graphs with radial layout.* In INFOVIS., pages 43–50, 2001. (Cité en page 80.)

[Yeung 2003] K.Y. Yeung. *Clustering or automatic class discovery : non-hiérarchical, non-SOM.* dans A pratical approch to microarray data analysis, Kluwer Acadamic Publisher, Boston., 2003. (Cité en page 49.)

[Younis 2004] O. Younis et S. Fahmy. *HEED : a hybrid, energy-efficient, distributed clustering approach for ad hoc sensor networks*. IEEE Transactions on Mobile Computing, vol. 3, no. 4, pages 366–379, 2004. (Cité en page 85.)

[Zhang 1996] Tian Zhang, Raghu Ramakrishnan et Miron Livny. *BIRCH : An Efficient Data Clustering Method for Very Large Databases*. In H. V. Jagadish et Inderpal Singh Mumick, editeurs, Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, Montreal, Quebec, Canada, June 4-6, 1996, pages 103–114. ACM Press, 1996. (Non cité.) :1996fc :1996fc :1996fc :1996fc

[Zighed 1999] D.A. Zighed et M. Sebban. Apprentissage automatique, chapitre Sélection et validation statistique de variables et de prototypes. Marc Sebban and Gilles Venturini, octobre 1999. (Cité en page 80.)

[Zighed 2002] D.-A. Zighed, S. Lallich et F. Muhlenbach. *Separability index in supervised learning. In Principles of Data mining and Knowledge Discovery*. In 6th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'02). Heidelberg, Germany. Springer. T. Elomaa, H. Mannila, H. Toivonen (eds), pages 475–487, 2002. (Cité en page 81.)

[Zighed 2005] Djamel-Abdelkader Zighed, Stéphane Lallich et Fabrice Muhlenbach. *A statistical approach of class separability*. Applied Stochastic Models in Business and Industry, vol. 21, no. 2, pages 187–197, 2005. (Cité en page 81.)

[Zighed 2006] Djamel-Abdelkader Zighed et Hakim Hacid. *Proximity graphs and separability of classes*. In 11th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU 06), Paris, France, pages 1488–1495. IPMU, July 2006. (Cité en page 81.)