Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université Laghrour Abbès Khenchela

Ecole nationale Supérieure d'Informatique (E.S.I)

École Doctorale

Sciences et **T**echnologies de l'**I**nformation et de la **C**ommunication

Mémoire

En vue de l'obtention du diplôme de

Magister en informatique

Option : Systèmes d'**I**nformations et de **C**onnaissances (SIC)

Contribution à la Fouille de Données : Clustering de Données basé sur les Colonies de Fourmis avec Contrôle de l'Emergence

Présenté par : **Billel KENIDRA**

Soutenu devant le jury:

Pr. Batouche Mohamed	Université Constantine 2	Président
Dr. Meshoul Souham	Université Constantine 2	Rapporteur
Dr. Melkemi Kamel Eddine	Université Biskra	Examinateur
Dr. Bouzenada Mourad	Université Constantine 2	Examinateur

ANNÉE UNIVERSITAIRE **2013/2014**

DÉDICACES

Je dédie ce modeste travail :

A toute ma famille

A tous mes amis

A tous ceux qui m'ont aidé

REMERCIEMENTS

Je remercie Mme MESHOUL Souham, Maître de conférence à l'Université de Constantine qui a dirigé et encadré ce mémoire. Je la remercié particulièrement pour la disponibilité, le regard critique, la compétence et la sagesse, avec lesquelles elle m'a guidé tout au long de ce travail.

Je remercie tous les membres du jury qui ont accepté d'évaluer mon travail.

Je tiens aussi à remercier tous ceux qui ont, de près ou de loin, aidé à rendre ce travail possible, que ce soit par des idées ou par des encouragements.

RÉSUMÉ

Le problème de clustering de données ou classification non supervisée a été identifié comme une des problématiques majeures en fouille de données ou datamining. Son but est d'identifier et d'extraire des groupes significatifs à partir d'un ensemble, souvent très large, de données définies par des valeurs d'attributs. Ce problème est de nature combinatoire car le nombre de partitions qu'il est possible d'avoir croît de façon exponentielle avec le volume de données à classer et le nombre de clusters. C'est la raison pour laquelle il est souvent traité sous l'angle de l'optimisation.

Par ailleurs, la nature est une immense source d'inspiration pour résoudre des problèmes complexes tels que le clustering des données, puisqu'elle est riche en systèmes naturels extrêmement divers, dynamiques, robustes et complexes. Dans ce contexte, les algorithmes inspirés de la nature sont des méta-heuristiques qui imitent des systèmes naturels, pour résoudre des problèmes d'optimisation.

Dans les systèmes complexes à organisation autonome, les effets émergents peuvent se produire sans qu'ils ne soient ni voulus, ni prévus dans la phase de conception. Pour rendre les systèmes fiables, il est nécessaire de prendre soin de ce problème. Le contrôle de l'émergence consiste à introduire des boucles de rétroaction pour rendre les systèmes auto-adaptatifs.

Dans le présent travail, nous présentons une nouvelle approche de clustering de données à des fins de datamining basé sur les colonies de fourmis avec contrôle de l'émergence par boucle de rétroaction. Cette approche est caractérisée par trois éléments essentiels à savoir :

- doter chaque fourmi d'une mémoire courte gérée selon une stratégie FIFO,
- modifier les opérations de ramassage et de dépôt des objets ou fonction Pick and Drop,
- introduire des agents anti-clustering pour détruire l'émergence négative. Ces agents agissent de façon inverse par rapport au comportement de fourmis artificielles.

L'approche proposée a été implémentée sur la plateforme multi agents NETLOGO et a été testée sur des jeux de données synthétiques et réels. Les résultats obtenus en utilisant une mesure de qualité interne et une mesure de qualité externe sont très prometteurs.

Mots-clés: Fouille de données, Optimisation, Approches bio-inspirées, Extraction de connaissances, clustering par colonies de fourmis, émergence contrôlée.

Sommaire

Liste des figures

Liste des tableaux

INTRODUCTION GÉNÉRALE

1. Contexte du projet

L'utilisation de plus en plus généralisée de l'informatique permet, aujourd'hui, aux entreprises de récolter et de mettre à leur disposition une masse de données importante. Cependant, on ne sait toujours pas comment les exploiter une fois qu'elles ont rempli leur rôle principal (facturation par exemple). Pourtant les propriétaires de ces données hésitent souvent à leur destruction pour au moins deux raisons. D'une part le coût de stockage, en effet, l'évolution des machines en termes de puissance et de stockage ont encouragé les sociétés à accumuler encore plus d'informations. D'autre part, ils supposent que ces données contiennent peut être des connaissances d'une grande valeur (tel que le comportement des clients).

En revanche, Il faut anticiper les changements et les nouveaux besoins des clients. Pour que cette anticipation soit efficace, il faut disposer de l'information pertinente. Cette dernière est issue de plusieurs systèmes hétérogènes, c'est pour ça il faut se doter d'un mécanisme fiable, fort et puissant pour donner un instrument efficace aux décideurs, en leur permettant de prendre la bonne décision, ce mécanisme est le processus de fouille de données plus communément connu sous le nom de Data Mining.

L'abondance de données avec le besoin de bons outils d'analyse de données ont été décrits par « riche en données mais pauvre en informations ». Par conséquence, des décisions importantes sont souvent prises par l'intuition des décideurs, simplement parce que les décideurs n'ont pas les outils pour extraire les connaissances précieuses embarquées dans les grandes quantités de données. Le data mining permet de faire l'économie du temps, des moyens et des efforts. Le cercle vertueux du data mining est de transformer les données en informations, les informations en décisions et les décisions…en bénéfices. Les données, sont celles qui sont produites par des capteurs (relevés de données météo), des caisses de supermarchés,… ; elles relèvent de l'observation directe d'un phénomène, sans interprétation. Les données sont des faits bruts et des chiffres qui n'ont pas de sens.

L'information est une donnée extraite, interprétée, filtrée ou formatée d'une certaine manière, pour donner un sens aux données. Les connaissances sont issues de la combinaison et de l'interprétation des informations. Elles en forment une synthèse abstraite dans une perspective déterminée. Par nature, les connaissances sont proches des humains et de leurs besoins, elles sont plus facilement utilisables par eux que les données. De plus, l'aspect synthétique des connaissances leur donne en général un caractère abstrait qui permet leur utilisation dans des domaines variés.

Les techniques de Fouille de Données permettent de découvrir des informations pertinentes cachées dans les données. Elles permettent par exemple de trouver les caractéristiques des clients les plus fidèles d'une banque, lorsqu'un client se présente, à une société de crédit, pour avoir un prêt, la société de crédit est devant un embarras, surtout pour les clients qu'elle ne connaît pas encore.

Va-t-elle accepter cette demande de prêt, ce qui est légitime pour toute banque, en vue d'accroître le profit ? Va-t-elle refuser cette demande pour ne pas risquer de tomber sur de mauvais payeur, dans de tels cas, ça sera une perte sèche pour la banque ? La décision pour la banque serait d'autant plus facile que si elle dispose d'outils lui permettant de catégoriser les clients selon leur profil. Pour ce faire, une plateforme nécessaire serait la classification non supervisée des données ou clustering des données [Hanoune et al, 98].

2. Problématique

A travers l'exemple précédent, on voit bien que le clustering des données est au cœur de tout processus de data mining. La création de groupes au sein d'un ensemble d'éléments est une opération omniprésente dans notre société. Les groupes peuvent posséder plusieurs significations, comme par exemple une signification sociale quand ils décrivent un ensemble de personnes qui partagent des caractéristiques communes. Il est naturel de faire appel aux groupes quand il s'agit de structurer, d'organiser ou de résumer un ensemble d'éléments. Ainsi, dans la vie de tous les jours, la notion de groupe est utilisée pour l'organisation et la description, comme par exemple des familles de plantes, des genres de musique, des groupes de produits, des groupes sanguins, etc.

Un groupe peut être défini de manière informelle comme un ensemble d'éléments qui sont rassemblés en raison d'une relation particulière entre ces éléments. La problématique consistant à former de tels groupes de manière automatique se pose dans de nombreux domaines. En marketing, on va chercher à regrouper des personnes ayant des comportements de consommation similaires pour cibler par exemple une campagne publicitaire. Dans l'étude des réseaux sociaux, on cherche à regrouper les différents membres du réseau pour y faire émerger des communautés. En biologie, on cherche à identifier des groupes de gènes ayant le même comportement pour mettre en place des thérapies géniques.

La complexité du clustering est fonction de la taille du volume des données ainsi que du nombre de clusters. En effet, le nombre de façons de regrouper n points dans k clusters est de l'ordre de $k^n/k!$. La complexité du clustering des données est inhérente au manque d'informations préalables sur les distributions des données [Forestier, 10]. Pour gérer cette complexité, une voie

serait l'utilisation de méta-heuristiques. Dans le contexte de notre travail on s'intéresse particulièrement aux méta-heuristiques basées sur les colonies de fourmis artificielles.

3. Approche et Motivations

Pour effectuer le clustering il existe plusieurs techniques issues de diverses disciplines scientifiques (statistiques, bio-inspirées, base de données, théorie des graphes) pour faire apparaître des corrélations cachées dans des gisements de données, afin de construire des modèles à partir de ces données. Les méta-heuristiques sont un ensemble de méthodes qui permettent de concevoir des algorithmes. Elles permettent de résoudre des problèmes d'optimisation auxquels les ingénieurs et les décideurs sont régulièrement confrontés. La majorité des problèmes d'extraction de connaissances peuvent s'exprimer comme des problèmes d'optimisation combinatoire. Or, de nombreux problèmes d'optimisation combinatoire sont NP-difficiles et ne pourront donc pas être résolus de manière exacte dans un temps "raisonnable" puisque la capacité de calcul des machines évolue linéairement alors que le temps nécessaire à la résolution de ces problèmes évolue exponentiellement. Lorsqu'on s'attaque à des problèmes réels, il faut se résoudre à un compromis entre la qualité des solutions obtenues et le temps de calcul utilisé. Les méta-heuristiques sont souvent inspirées par des systèmes naturels, qu'ils soient pris en biologie de l'évolution (cas des algorithmes génétiques et les réseaux de neurones) ou encore en éthologie (cas de l'optimisation par essaims particulaires ou des algorithmes de colonies de fourmis).

Notre approche est basée sur ce dernier, nous avons adopté une méta-heuristique basée sur les colonies de fourmis. Ce choix tire sa force des avantages suivants :

- **L'optimisation de l'effort :** la coopération et la coordination indirecte entre les fourmis, créent une émergence de haut niveau (solution). Cette émergence est à la base des règles simples effectuées par chaque fourmi pendant une durée de temps.
- **L'optimisation du temps :** les approches basées sur les colonies de fourmis procèdent à se résoudre à un compromis entre la qualité des solutions obtenues et le temps de calcul utilisé.
- **La robustesse :** la non dépendance entre les fourmis, rends le système robuste, même un élément tombe en panne, le système reste en service.

Dans le cadre de ce travail, le clustering de données est traité sous l'angle de l'optimisation. Plus précisément, on propose une approche basée sur les colonies de fourmis où les données à regrouper sont vues comme des objets à ramasser par des fourmis. Cette approche exploite trois idées de base. La première concerne l'association d'une mémoire courte à chaque fourmi pour lui permettre

de se rappeler des derniers objets rencontrés dans son parcours. La seconde est liée à la sémantique accordée à la fonction de ramassage des objets plus communément connue dans la littérature par la fonction Pick and Drop Enfin, la troisième idée consiste à introduire des agents anti clustering afin de diversifier la recherche et échapper aux optima locaux. Cette dernière idée permet la mise en œuvre d'un mécanisme de contrôle d'émergence.

4. Organisation du mémoire

Ce document est structuré en cinq chapitres et se termine par une conclusion générale et trois annexes dont voici le contenu :

Chapitre 1 : dresse un panorama des techniques principales pour regrouper des données. Il précise l'objectif et les enjeux du clustering ainsi que les différentes approches existantes pour l'évaluation de résultats de clustering.

Chapitre 2 : présente un état de l'art du clustering par approches inspirées de la nature et notamment le clustering par fourmis artificielles.

Chapitre 3 : s'intéresse à l'opération du contrôle d'émergence et plus particulièrement à l'émergence contrôlée de l'essaim dans le processus du clustering basé sur les fourmis.

Chapitre 4 : décrit l'approche proposée dans le cadre de ce travail en mettant en exergue les idées exploitées et présente l'algorithme développé.

Chapitre 5 : décrit l'étude expérimentale réalisé et fournit les résultats obtenus en utilisant l'approche proposée ainsi qu'une étude comparative avec d'autres méthodes de la littérature. La fin de ce chapitre mets l'accent sur l'apport de notre approche dans le processus de data-mining.

Conclusion générale et perspectives : cette partie conclue le mémoire en présentant le bilan général de celui-ci ainsi que les perspectives. Cette conclusion nous a permis de lister quelques perspectives de prolongement de nos travaux.

Annexes : englobe trois annexes qui sont, un descriptif des jeux de données test utilisés, description et fonctionnalités du NetLogo (plateforme utilisée) et finalement un bilan détaillé des résultats obtenus sur plusieurs exécutions de chaque algorithme impliqué dans l'opération de comparaison.

CHAPITRE I: LE CLUSTERING DES DONNÉES

I.1 Introduction

La création de groupes au sein d'un ensemble d'éléments est une opération omniprésente dans notre société. Les groupes peuvent posséder plusieurs significations, comme par exemple une signification sociale quand ils décrivent un ensemble de personnes qui partagent des caractéristiques communes. Il est naturel de faire appel aux groupes quand il s'agit de structurer, d'organiser ou de résumer un ensemble d'éléments. Ainsi, dans la vie de tous les jours, la notion de groupe est utilisée pour l'organisation et la description, comme par exemple des familles de plantes, des genres de musique, des groupes de produits, des groupes sanguins, etc (Figure I.1).

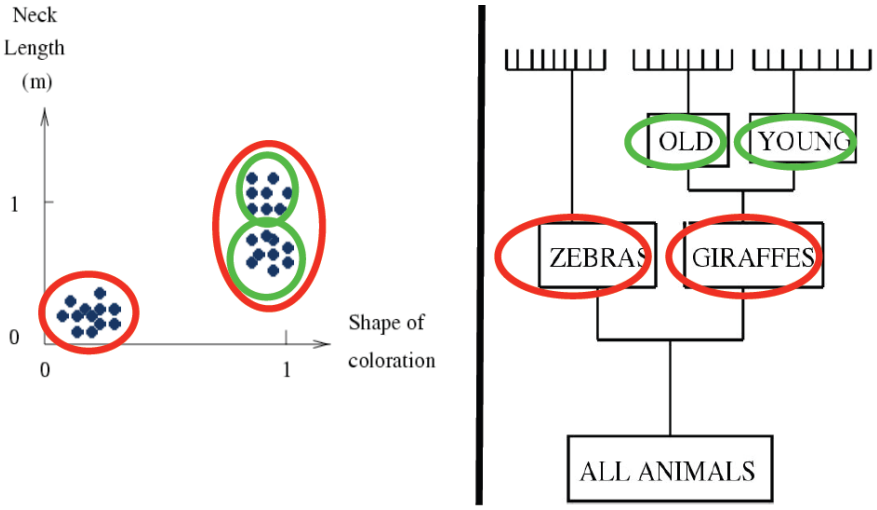
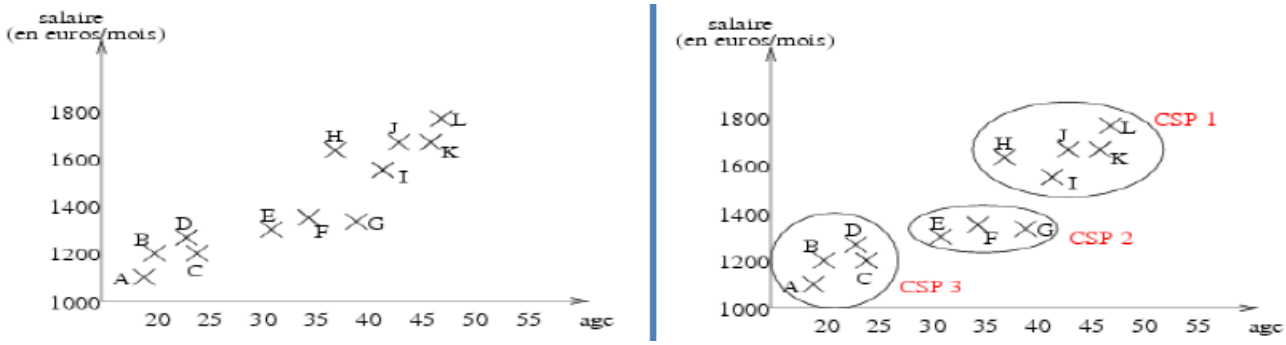


Figure I.1 Exemple illustratif pour le clustering

Un groupe peut être défini de manière informelle comme un ensemble d'éléments qui sont rassemblés en raison d'une relation particulière entre ces éléments. La problématique consistant à former de tels groupes de manière automatique se pose dans de nombreux domaines. En marketing, on va chercher à regrouper des personnes ayant des comportements de consommation similaires pour cibler par exemple une campagne publicitaire. Dans l'étude des réseaux sociaux, on cherche à regrouper les différents membres du réseau pour y faire émerger des communautés. En biologie, on cherche à identifier des groupes de gènes ayant le même comportement pour mettre en place des thérapies géniques.

Faire émerger des groupes au sein d'un ensemble d'éléments sans aucune information a priori est aussi appelée classification non supervisée ou classification automatique dans le sens où pas de classes prédéfinies. Les groupes créés sont appelés clusters. L'objectif du clustering est de découvrir la structure sous-jacente des données pour en extraire de l'information. C'est un concept intuitif mais

difficile à réaliser dans la pratique. Cette difficulté est causée par un manque de définition unique et précise d'un cluster dû à un manque d'informations préalables sur les distributions des données.



(a) un exemple d'ensemble d'individu à classer

(b) et le clustering résultat attendu

Figure I.2 Le clustering sur un exemple [Devèze and Fouquin, 05]

Le clustering facilite donc l'étude d'une population d'effectif important on les regroupe en plusieurs classes de telle sorte que les individus d'une même classe soient le plus semblables possible - au regard d'une certaine métrique- (homogénéité intra-classe), et que les classes soient le plus distinctes possibles (hétérogénéité inter-classe). La mise en œuvre des méthodes de clustering a pour but de regrouper les concepts similaires ensemble et de les projeter sur une carte conceptuelle interprétable [Devèze and Fouquin, 05].

Dans ce chapitre la place occupée par le clustering dans un processus de data mining sera mise en exergue et les concepts de base et les techniques principales de clustering seront présentées. Dans cet état de l'art, nous présentons les algorithmes les plus utilisés, les plus connus, et ceux qui exhibent des idées différentes et intéressantes. A la fin, nous terminons par donner différents types de mesures de validité de clusters.

I.2 Le clustering en data mining

Les algorithmes de clustering visent à segmenter une population hétérogène en un certain nombre de sous-groupes plus homogènes, ou clusters, tout en maximisant l'homogénéité à l'intérieur de chaque groupe et en la minimisant entre ces derniers [Larose, 05].

I.3 Les trois principales étapes du clustering

Le processus de clustering se divise en trois étapes majeures (figure I.3) : (1) la préparation des données, (2) l'algorithme de clustering et (3) l'exploitation des résultats de l'algorithme. Nous discutons ici des problématiques générales liées à chacune de ces étapes [Cleuziou, 04].

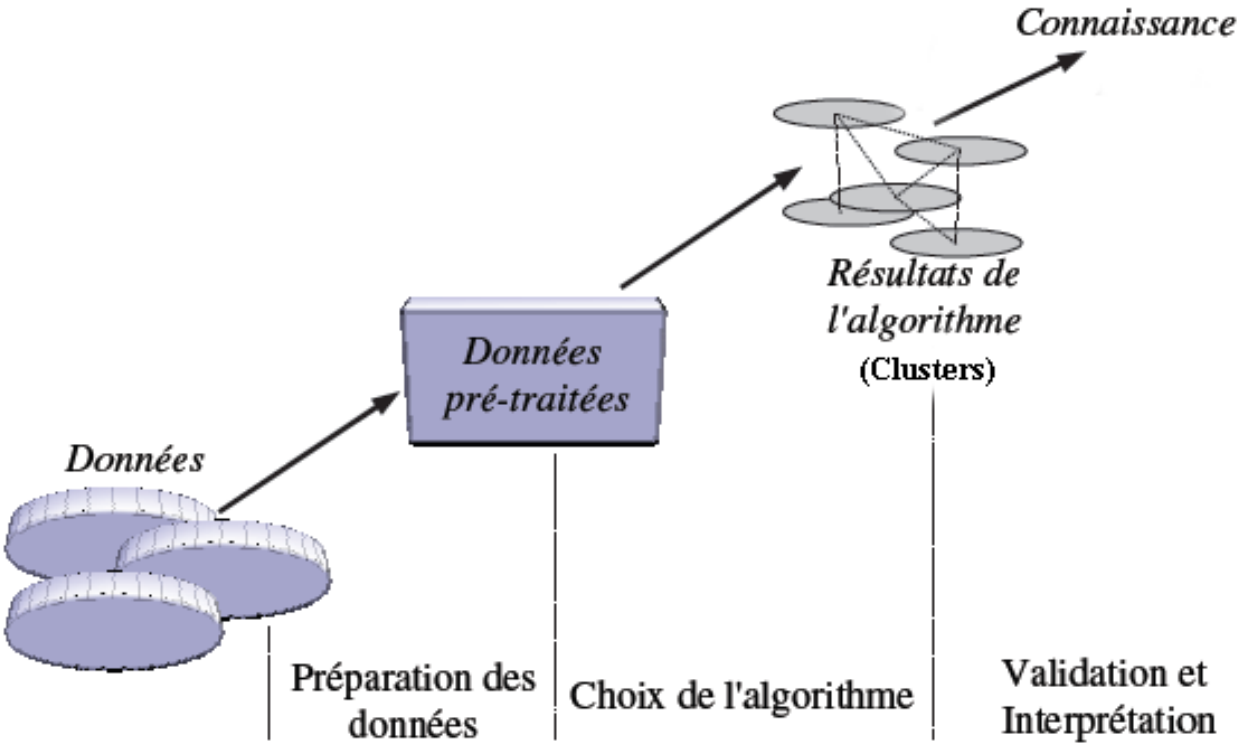


Figure I.3 Les différentes étapes du processus de clustering [Cleuziou, 04]

I.3.1 La préparation des données

La préparation des données (prétraitement) est définie par l'ensemble des opérations qui permettent de convertir les données brutes en données préparées et adaptées à la méthode de fouille envisagée. Les opérations de préparation peuvent être de différents types : l'intégration des données, le nettoyage des données, la réduction des données, la transformation des données. Un processus de prétraitement de données peut être décrit selon les étapes suivantes :

I.3.1.1 La sélection de données

Cette phase concerne le filtrage de données, c'est à dire la réduction de la dimension des données (élimination d'attributs sans intérêt, ou ayant beaucoup de valeurs erronées et manquantes), ainsi que la réduction la taille des données (des enregistrements). Une deuxième réduction peut être faite par des techniques statistiques d'échantillonnage, s'il s'avère que la masse de données est trop grande, et que l'application du clustering serait ainsi très coûteuse en terme de temps CPU et d'espace mémoire.

I.3.1.2 Le nettoyage et l'intégration et des données

Cette tâche consiste à détecter et à supprimer les erreurs, le bruit et l'incohérence de données pour améliorer l'intégration de données de sources multiples.

I.3.1.3 La transformation de données

Plusieurs algorithmes de clustering sont contraignants sur la forme des données qu'ils traitent. Cette étape consiste à préparer les données brutes et à les convertir en données appropriées. La transformation se fait par attribut (toutes les valeurs d'un attribut doivent être transformées).

La discrétisation de variables continues est un exemple de transformation d'attributs. Il s'agit de transformer un attribut continu en divisant son domaine en intervalles finis. Ainsi, le domaine de l'attribut transformé devient un ensemble de valeurs discrètes.

L'agrégation de données est un autre type de transformation. L'agrégat d'un attribut est la transformation de ce dernier par une règle ou équation. Imaginons que l'ont veut analyser les salaires annuels des employés, et que l'on dispose seulement des salaires mensuels. Un nouvel attribut agrégat serait le salaire multiplié par douze.

I.3.2 Le choix de l'algorithme

Le choix de l'algorithme de clustering doit donner lieu à une analyse globale du problème : quelle est la nature (qualitative et quantitative) des données ? Quelle est la nature des clusters attendus (nombre, forme, densité, etc.) ? L'algorithme doit alors être choisi de manière à ce que ses caractéristiques répondent convenablement à ces deux dernières questions. Les critères de décision peuvent être : la quantité de données à traiter, la nature de ces données, la forme des clusters souhaités ou encore le type de schéma attendu (pseudo-partition, partition stricte, dendrogramme, etc.).

I.3.3 L'exploitation des clusters (interprétation des résultats)

L'étape d'exploitation des clusters est également appelée l'interprétation des résultats. C'est à l'analyste de transformer et d'interpréter les résultats bruts de clusters en unités de connaissance plus directement exploitables pour l'utilisateur final. L'analyste peut alors être amené à filtrer parmi les connaissances extraites celles qu'il juge triviales, redondantes, sans intérêt, fausses en comparaison de ce qu'il souhaite trouver. De même, cette étape d'analyse permet d'envisager le recours à une autre approche de clustering plus adaptée [Cleuziou, 04].

I.4 Mesures de similarité

Les résultats d'une technique de clustering dépendent fortement de la mesure de similarité choisie, cette mesure doit être choisie avec prudence. En effet la mesure de similarité repose sur le calcul de la distance entre deux données, sachant que chaque donnée est composée d'un ensemble

d'attributs quantitatifs et/ou qualitatifs. Plus la distance est importante, moins similaires sont les données et vice versa.

I.4.1 Types et échelles de données

La mesure de proximité et le type de clustering utilisés dépendent des types et des échelles des attributs de données [Jain, 88]. Une variable quantitative peut être *binaire*, *continue* ou *discrète,* voir le Tableau I.1, tandis que les différentes échelles de données sont montrées dans le Tableau I.2.

Binaire	Les variables *binaires* ne peuvent prendre que deux valeurs, {0,1}, {succès, échec}, {absence, présence} ou {Masculin, Féminin}…
Discret	Les variables *discrètes* sont celles dont les valeurs forment un sous-ensemble fini ou infini de l'ensemble ***N*** des entiers naturels (*exemples* : le nombre de jours d'hospitalisation, le nombre d'enfants).
Continu	Les variables *continues* ou *d'échelle* sont les variables dont les valeurs forment un sous-ensemble infini de l'ensemble ***R*** des réels (*exemple* : le salaire, le coût du séjour)

Tableau I.1 Les différents types d'attributs [Kumar, 00]

Qualitative (*catégorielle*)	Nominal	Les valeurs sont juste des noms différents. par exemple: les codes postaux, les couleurs, le sexe.
	Ordinal	Les valeurs reflètent un ordre, rien plus. par exemple : bon, meilleur, mieux ou couleurs ordonnées par le spectre.
Quantitative	Intervalle	La différence entre les valeurs est significative par exemple, l'intervalle de température.
	Ratio	Par exemple la hauteur, la largeur et la longueur d'un objet.

Tableau I.2 Les différentes échelles de données [Kumar, 00]

I.4.2 Distance et similarité

Une bonne méthode de regroupement permet de garantir une grande similarité intra-groupe et une faible similarité inter-groupe. La qualité d'un regroupement dépend de la mesure de similarité utilisée par la méthode et de son implémentation [Pedrycz, 05]. Le calcul de la similarité permet d'estimer à quel point deux points sont proches, et sur la base de cette proximité se fait l'affectation des objets aux clusters. Formellement, la similarité $d(x,y)$ entre x et y est considérée comme une fonction satisfaisant les conditions suivantes : $d(x,y) \geq 0$ $d(x,x) = 0$ $d(x,y) = d(y,x)$

Fonction de distance	Formule
Distance Euclidienne	$d(x,y)=\sqrt{\sum_{i=1}^{n}(x_i-y_i)^2}$
Distance de Hamming	$d(x,y)=\sum_{i=1}^{n}\lvert x_i-y_i\rvert$
Distance de chebyschev	$d(x,y)=\max_{i=1,2,\ldots,n}\lvert x_i-y_i\rvert$
Distance de Minkowski	$d(x,y)=\sqrt[p]{\sum_{i=1}^{n}(x_i-y_i)^p}, p>0$
Distance de Canberra	$d(x,y)=\sum_{i=1}^{n}\frac{\lvert x_i-y_i\rvert}{x_i+y_i}, x_i \text{ et } y_i \text{ sont positifs}$
Séparation angulaire	$d(x,y)=\frac{\sum_{i=1}^{n}x_i y_i}{\left[\sum_{i=1}^{n}x_i^2\sum_{n=1}^{n}y_i^2\right]^{1/2}}$

Tableau I.3 Fonctions de distance entre deux points x et y [Pedrycz, 05]

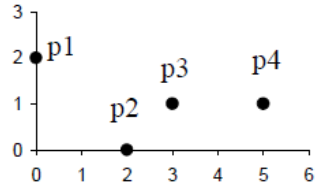
La distance est la mesure la plus utilisé parmi les types de mesures de similarité et de dissimilarité. Le Tableau I.3 présente quelques fonctions de distance.

I.4.3 La matrice de données

Les objets (échantillons, mesures, modèles, événements) sont habituellement représentés comme des points (vecteurs) dans un espace multidimensionnel, où chaque dimension représente un attribut distinct (variable, mesure) décrivant l'objet. Ainsi, un ensemble d'objets est représenté comme une matrice *m x n*. Cette matrice est appelée matrice de données ou jeu de données. La Figure I.3, ci-dessous, fournit un exemple correspondant.

I.4.4 La matrice de proximité

Plusieurs algorithmes de clustering utilisent la matrice de données originale et beaucoup d'autres utilisent une matrice de similarité, ou une matrice de dissimilarité. Pour la convenance, les deux matrices sont généralement mentionnées comme une matrice de proximité, qui est une matrice de *m x m* contenant toutes les dissimilarités ou les similarités entre les objets considérés. Si p_i et p_j sont le $i^{ème}$ et le $j^{ème}$ objets, respectivement, alors l'entrée à la $i^{ème}$ ligne et la $j^{ème}$ colonne de la matrice de proximité est la similarité, ou la dissimilarité, entre l'objet p_i et l'objet p_j.



point	x	y
p1	0	2
p2	2	0
p3	3	1
p4	5	1

	p1	p2	p3	p4
p1	0.000	2.828	3.162	5.099
p2	2.828	0.000	1.414	3.162
p3	3.162	1.414	0.000	2.000
p4	5.099	3.162	2.000	0.000

Figure I.4 Quatre points, leur matrice de données et leur matrice de proximité [Kumar, 00]

La figure I.3 montre, respectivement, quatre points, leur matrice de données et leur matrice proximité correspondante, via l'utilisation de la distance euclidienne comme une fonction de distance.

I.4.5 Mesure de ressemblance entre individus à descriptions symboliques

La présentation de la section précédente illustre la diversité des mesures existantes et l'importance du choix de la distance ou de la similarité dans le processus de classification automatique pour ne pas trop influencer son déroulement. Le choix de la dissimilarité/similarité est facilité lorsqu'on est en présence d'un type unique de données. Néanmoins dans les applications réelles, il est courant d'avoir affaire à des données de différents types dites *hétérogènes* ou *mixtes*. Ce qui se traduit par le fait que la classification porte sur un *« Tableau de données symboliques ou complexes»* contenant à la fois des variables *univaluées* (*quantitatives* ou *qualitatives*) et *multivaluées* (nous nous limiterons ici à un *ensemble de modalités*). Dans ce cas, les mesures de proximités usuelles entre deux individus ne sont pas directement applicables. Pour répondre à ce problème, il existe une stratégie qui est:

L'homogénéisation de la matrice de données : il s'agit de transformer les variables pour les homogénéiser (afin qu'elles aient en fin le même type), puis utiliser une fonction de comparaison globale qui tient compte de toutes les variables afin de calculer un indice de proximité entre les individus. *Dans le cas classique*, où les individus sont décrits à la fois par des variables *univaluées* (quantitatives et qualitatives), plusieurs opérations de conversion qui permettent de passer d'un type à un autre sont définies dans la littérature. De telles transformations étant réalisées, la matrice de données devient homogène et nous retombons dans le cadre classique.

Dans le cas symbolique, où les individus sont décrits à la fois par des variables *univaluées* (quantitatives et qualitatives) et *multivaluées*, il ne s'agit pas de transformer les descriptions multivaluées en une modalité unique afin de ne pas perdre l'information contenue dans ces descriptions, mais plutôt de passer d'une description univaluée à une description multivaluée. Par exemple une valeur *V* peut être vue comme l'intervalle [*V*, *V*] dans le cas d'une variable quantitative, ou bien l'ensemble {*V*} dans le cas d'une variable qualitative [Chavent, 97]. Ces différentes

transformations permettent d'obtenir une matrice homogène où toutes les descriptions sont multivaluées.

La distance la plus utilisée pour les données de type nominale est la distance de Hamming. Soit deux objets a et b décrits par *n* variables nominales, la distance de Hamming peut être définie de la manière suivante [Jourdan, 03] :

$$d_H(a,b) = \sum_{i=1..n} d(a_i, b_i) \text{ avec } d(a_i, b_i) = \begin{cases} 1 \text{ si } a_i \neq b_i \\ 0 \text{ sinon} \end{cases}$$

I.5 Techniques principales de Clustering

Une multitude de techniques de clustering est développée. Les techniques peuvent différer dans leurs principes, dans leurs propriétés, dans leurs paramètres et dans leurs formes générales du partitionnement généré. La catégorisation de ces techniques peut être réalisée selon plusieurs aspects : la mesure de proximité (similarité/dissimilarité) utilisée entre les données, la théorie ou les concepts fondamentaux sur lesquels se basent les techniques, la nature des données manipulées et beaucoup d'autres critères. En général, un algorithme de clustering nécessite:

- Le choix d'une mesure de proximité appropriée au domaine des données.
- La définition d'un critère de clustering qui peut être exprimé par l'intermédiaire d'une fonction objectif ou d'un autre type de règles.
- Une représentation simple et compacte de l'ensemble de données et l'ensemble des clusters.
- Le clustering : cette étape peut être réalisée de nombreuses façons, selon la technique adoptée.
- La validation des résultats : la qualité des résultats d'un algorithme de clustering est vérifiée en utilisant des mesures de validation de clusters. La qualité de chaque cluster doit être jugée non seulement par la fonction objectif, mais aussi selon un critère d'évaluation externe.
  [Law et al, 04]

Les algorithmes peuvent être classifiés de façon générale dans les catégories suivantes [Rogovschi, 09]:

- Clustering par partitionnement.
- Clustering hiérarchique.
- Clustering basé sur la densité.
- Clustering basé sur les grilles.

Pour chacune de ces catégories, il y a beaucoup d'algorithmes différents pour trouver les clusters.

I.5.1 Clustering par partitionnement

Les approches de classification par partitionnement permettent de subdiviser l'ensemble des individus en un certain nombre de classes en employant une stratégie d'optimisation itérative dont le principe général est de générer une partition initiale, puis de chercher à l'améliorer en réattribuant les données d'une classe à l'autre [Elghazel, 07].

Les techniques par partitionnement sont divisées en deux sous-catégories principales [Kotsiantis et al, 01], les algorithmes basés sur les centroïdes et les algorithmes basés sur les médoïdes. Nous allons décrire les deux algorithmes les plus connus : Kmeans et Kmedoid.

Ces deux techniques sont basées sur l'idée qu'un point de centre peut représenter un cluster. Pour Kmeans on utilise la notion du centroïde qui est le point de la moyenne ou la médiane d'un groupe de points. Pour Kmedoid on utilise la notion d'un médoïde qui est le point le plus représentatif d'un groupe de points.

A. Kmeans : L'algorithme des K-moyennes est dédié aux tâches de segmentation. L'objectif est de segmenter les données en k groupes, k étant fixé a priori. On part de K données synthétiques que l'on nomme des « centres ». Chaque centre caractérise un groupe. À chaque centre sont associées les données qui lui sont les plus proches, cela crée un groupe autour de chaque centre. Ensuite, on calcule le centre de gravité de chacun de ces groupes, ces k centres de gravité deviennent les nouveaux centres et on recommence tant que les groupes ne sont pas stabilisés, i.e. tant qu'il y a des données qui changent de groupe d'une itération à la suivante [Ganaoui and Perrot, 04].

L'algorithme K-means est en 4 étapes :

1. Choisir K objets formant ainsi K clusters
2. (Ré) affecter chaque objet O au cluster Ci de centre Mi tel que dist(O,Mi) est minimal
3. Recalculer Mi de chaque cluster (le nouveau centre)
4. Aller à l'étape 2 s'il faut faire une affectation

Exemple illustratif : Supposons que nous avons huit points dans un espace bidimensionnel comme illustrés dans la Figure I. 4 et nous sommes intéressés à deux clusters (K = 2).

a	*b*	*c*	*d*	*e*	*f*	*g*	*h*
(1,3)	(3,3)	(4,3)	(5,3)	(1,2)	(4,2)	(1,1)	(2,1)

Tableau I.4 Points et leurs coordonnées

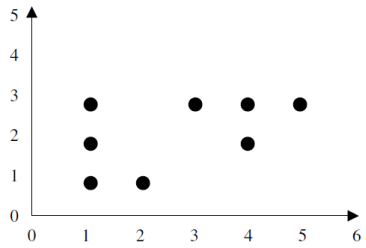


Figure I.5 Comment K-means partitionne ces points en 2 clusters

1. On choisit deux clusters C_1 et C_2 on affecte aléatoirement les points *g* et *h*, aux centres des clusters C_1 et C_2 respectivement. On obtient m1 = (1,1) et m2 = (2,1).
2. En utilisant la distance euclidienne entre chaque point et chaque centre m1 et m2 pour affecter chaque point au cluster ayant le centre le plus près, comme monté dans le Tableau I.5 :

Point	Distance de m1	Distance de m2	Est affecté au Cluster
a	2.00	2.24	C_1
b	2.83	2.24	C_2
c	3.61	2.83	C_2
d	4.47	3.61	C_2
e	1.00	1.41	C_1
f	3.16	2.24	C_2
g	0.00	1.00	C_1
h	1.00	0.00	C_2

Tableau I.5 Affectation des points au cluster approprié

Le cluster 1 contient les points {a,e,g}, et le cluster 2 contient les points {b,c,d,f,h}.

3. Pour chaque cluster, on calcule le nouveau centre : m1 devient [(1 + 1 + 1) /3, (3 + 2 + 1) /3] = (1,2). Et m2 devient [(3 + 4 + 5 + 4 + 2) /5, (3 + 3 + 3 + 2 + 1) /5] = (3.6, 2.4).

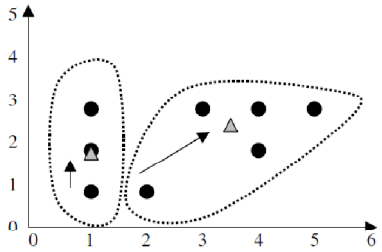


Figure I.6 Les clusters et leurs centres après la 1ière itération

4. On répète les étapes 2 et 3, mais cette fois avec les nouveaux centres m1 = (1,2) et m2 = (3.6, 2.4). On obtient :

Point	Distance de m1	Distance de m2	Est affecté au Cluster
a	1.00	2.67	C_1
b	2.24	0.85	C_2

c	3.16	0.72	C_2
d	4.12	1.52	C_2
e	0.00	2.63	C_1
f	3.00	0.57	C_2
g	1.00	2.95	C_1
h	1.41	2.13	C_1

Tableau I.6 Réaffectation améliorée

Il y a eu un changement d'un seul point (*h*) de groupe 2 au groupe 1. Le cluster 1 devient {a,e,g,h}, et le cluster 2 devient {b,c,d,f}. Pour chaque cluster, on calcule le nouveau centre :

m1 devient [(1 + 1 + 1 + 2)/4, (3 + 2 + 1 + 1)/4] = (1.25, 1.75)

m2 devient [(3 + 4 + 5 + 4)/4, (3 + 3 + 3 + 2)/4] = (4, 2.75)

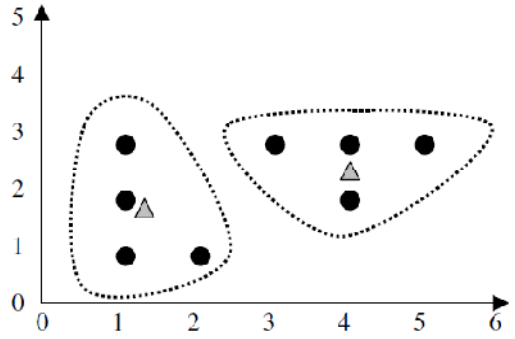


Figure I.7 Les clusters et leurs centres après la 2ième itération

Après la 3^{ième} itération, il s'est avéré qu'aucun changement des points par rapport aux clusters n'a eu lieu, donc on s'arrête [Larose, 05].

B. Kmedoid : Dans l'approche Kmedoid, un cluster est représenté par un de ses points. Ce point représentatif est appelé médoïde, c'est un point qui est le plus placé au centre en tenant en compte quelques mesures, comme par exemple, la distance. Le $i^{ème}$ médoïde est calculé en utilisant $\sum_{j=1}^{ni} Pij$ où Pij est la proximité entre le $i^{ème}$ médoïde et le $j^{ème}$ point dans le cluster. Pour des raisons de similarités (dissimilarités) on veut que cette somme soit le plus possible grande (petite). Cela est très coûteux en temps de calcul, car la découverte d'un meilleur médoïde exige l'essai de tous les points qui ne sont pas médoïdes.

L'algorithme est conceptuellement simple, il est décrit comme suit [Kumar, 00]:

1) Choisir *k* points initiaux. Ces points sont les médoïdes candidats qui sont destinés à être les points les plus centraux de leurs clusters (points représentatifs).

2) Considérer l'effet de remplacer un des points choisis (médoïdes) avec un des points non choisis. Conceptuellement, ceci est fait de la façon suivante :

   On calcule la distance entre chaque point non choisi et le médoïde candidat le plus proche et on calcule la somme de toutes les distances, cette somme représente le "coût" de la configuration actuelle. Tous les échanges possibles d'un point non choisi par un autre choisi sont considérés, et le coût de chaque configuration est calculé.
3) Choisir la configuration avec le coût le plus bas. Si c'est une nouvelle configuration, alors répéter l'étape 2.
4) Sinon, associer chaque point non choisi au point choisi le plus proche (médoïde) et arrêter.

Les méthodes kmedoid présentent l'avantage d'être applicable à tout type de données et sont dans l'ensemble plus robustes aux points aberrants que les méthodes des kmeans.

Plusieurs algorithmes basés sur la notion de médoïde existent comme PAM (Partitioning Around Medoids) qui est un algorithme Kmedoid qui essaye de grouper un ensemble de *m* points en k clusters en exécutant les étapes décrites ci-dessus. CLARA (Clustering LARge Applications) est une adaptation de PAM pour manipuler de grands jeux de données. CLARANS est né de deux algorithmes de clustering PAM et CLARA. CLARANS est un des premiers algorithmes de clustering qui a été développé précisément pour le datamining spatial [Rogovschi, 09].

I.5.2 Clustering hiérarchique

Les méthodes hiérarchiques construisent une hiérarchie de clusters, c'est-à-dire un arbre de clusters pouvant se présenter sous la forme d'un dendrogramme (Figure I.9). Ce dendrogramme décrit l'ordre dans lequel les points sont fusionnés (vue de bas en haut) ou les clusters sont fractionnés (vue de haut en bas).

Les méthodes de clustering hiérarchique sont décomposées en deux types d'approches, les approches ascendantes (agglomératives) et les approches descendantes (divisives). Dans les approches ascendantes les classes les plus similaires sont fusionnées deux à deux jusqu'à obtenir une seule classe contenant tous les éléments. Les approches divisives (descendantes) procèdent de manière inverse (Figure I.8). [Forestier, 10].

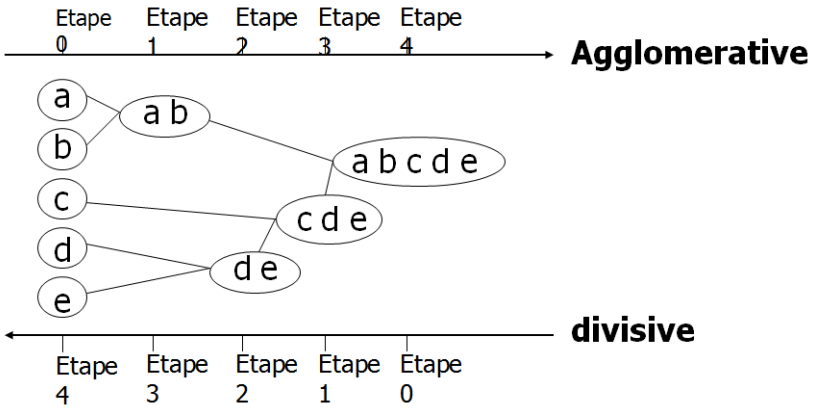


Figure I.8 Approche agglomérative & approche divisive [Han et al, 06]

Le dendrogramme peut être coupé à différents niveaux (niveau de similarité) pour donner les différents clusters des données.

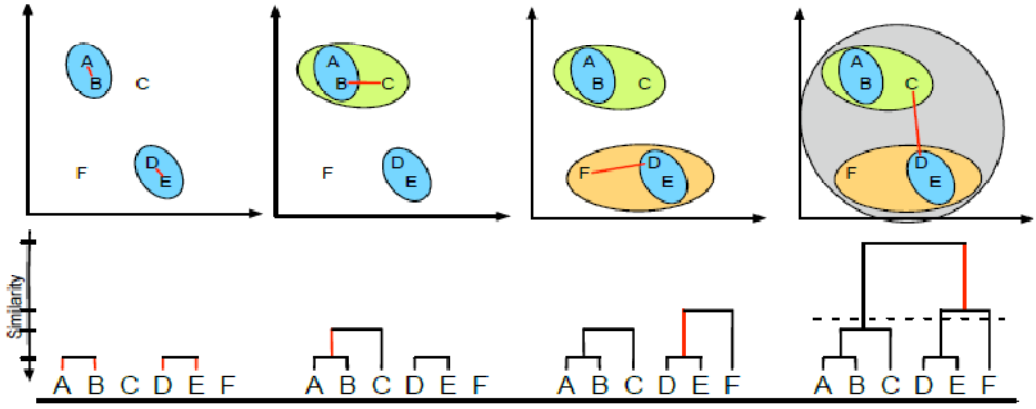


Figure I.9 Dendrogramme d'une méthode hiérarchique ascendante [Attar, 12]

La fusion ou la division des clusters dans les méthodes hiérarchiques dépend directement du critère de (dis)similarité utilisé entre les classes. Pour choisir les clusters à fusionner ou à diviser, une matrice de distance des clusters deux à deux est utilisée. Il existe plusieurs stratégies pour calculer la similarité entre les clusters, les plus connues étant : *single-link*, *complete-link* et *average-link*. La stratégie *single-link* compare les deux clusters en considérant la distance minimale entre les objets des deux clusters [Forestier, 10] :

$$D_s(C_1, C_2) = \min_{x \in C_1, y \in C_2} d(x, y)$$

La stratégie *complete-link* considère la distance maximale entre les objets des deux clusters :

$$D_c(C_1, C_2) = \max_{x \in C_1, y \in C_2} d(x, y)$$

Enfin, la stratégie *average-link* considère la moyenne des distances des objets des deux clusters :

$$D_a(C_1, C_2) = \frac{1}{|C_1||C_2|} \sum_{x \in C_1} \sum_{y \in C_2} d(x, y)$$

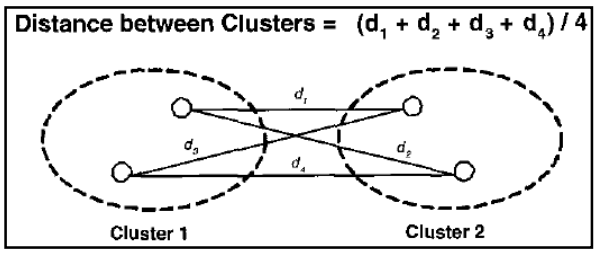


Figure I.10 Exemple illustratif de la stratégie average-link [Berry et al, 06]

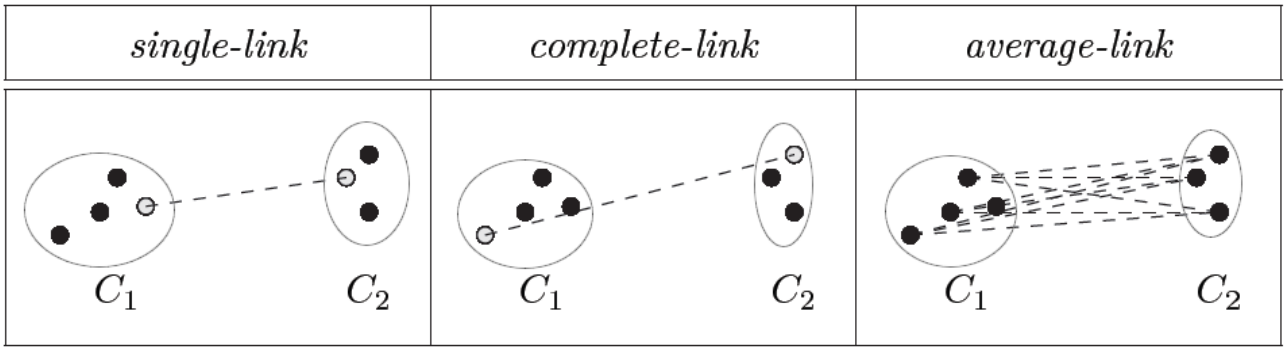
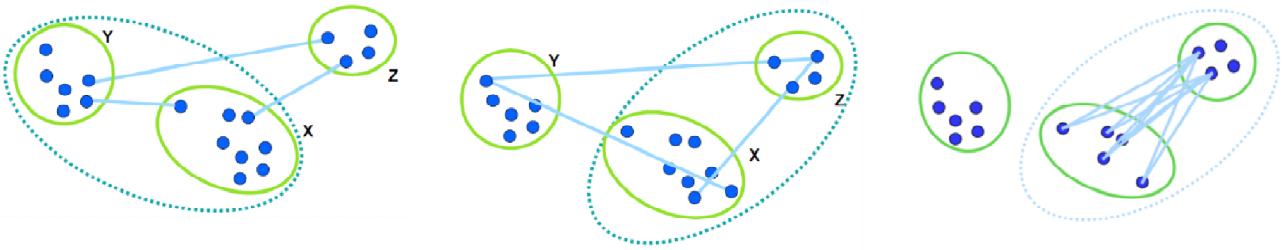


Figure I.11 Les différentes stratégies de regroupement de clusters en clustering hiérarchique [Forestier, 10]

C.-à-d. le choix de cluster le plus proche se diffère selon la stratégie adoptée (Figure I.11).



single-link *complete-link* *average-link*

Figure I.12 Le choix de cluster le plus proche selon la stratégie adoptée [Carey et al, 07]

I.5.3 Clustering basé sur la densité

Les algorithmes basés sur la densité considèrent les clusters comme des régions denses séparées par des régions qui le sont moins. [Halkidi et al, 01a]. Un des algorithmes les plus connus de cette catégorie est le DBSCAN. Son idée principale consiste à calculer le voisinage de chaque objet, c'est-à-dire les objets situés à une certaine distance de celui-ci. Si un objet possède un nombre important de voisinage, alors il est considéré comme appartenant à une région dense, sinon, il est déclaré comme une donnée atypique. DBSCAN nécessite la déclaration de deux paramètres principaux qui sont : le rayon de voisinage autour des points **Epsilon** (**ε**), et le nombre minimal de points qui doit être trouvé dans ce voisinage **MinPts**, ce qui permet d'établir un seuil de densité. L'algorithme opère de la façon suivante pour chaque point du nuage :

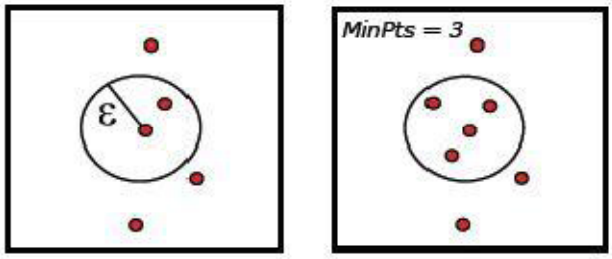


Figure I.13 Clustering basé sur la densité [Mayers, 10]

Prenant un point au hasard, l'algorithme crée un cluster à partir de ce point (on dit que c'est le noyau) s'il réalise ces deux conditions : il faut qu'il y ait au moins un certain nombre de points, déterminé par **MinPts**, à une distance de voisinage inférieure à **ε**. Sous la Figure I.12, un cluster est formé à droite mais pas à gauche pour un **MinPts** de 3. Si ce point ne réalise pas ces conditions, il est considéré comme bruit.

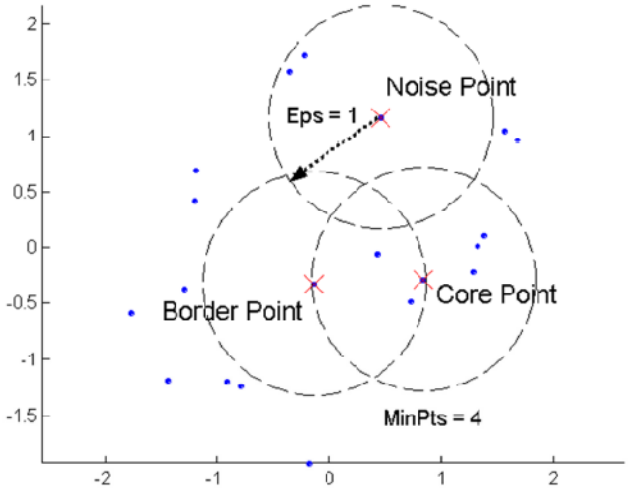


Figure I.14 Les types de points dans le clustering basé sur la densité [Mayers, 10]

Remarque :

- La densité est le nombre de points à l'intérieur d'un rayon donné (**ε**)
- Un point est un noyau s'il a plus qu'un nombre donné de points (MinPts) à l'intérieur d'un rayon donné (**ε**).
- Un point frontière est un point non noyau qui a un nombre de points inférieur à MinPts à l'intérieur de son (**ε**).
- Le bruit est l'ensemble de points non contenus dans n'importe quel cluster.
- L'algorithme commence par un point au hasard, mais le résultat trouvé sera toujours le même.
- L'approche générale de la segmentation basée sur la densité est de:
    - Trouver les points ayant un grand ensemble de voisinage.

- Trouver les voisins de ces points et les regrouper dans des segments.

- Deux clusters de densité différente peuvent fusionner si la distance qui les sépare est inférieure à **ε**. C'est pour cela que les paramètres **ε** et **MinPts** sont déterminants pour le nombre de clusters trouvé. On peut aussi noter que le choix de ces deux paramètres permet de supprimer le bruit plus ou moins efficacement. [Mayers, 10]

Plusieurs versions DBSCAN ont été proposées. Une version incrémentale de DBSCAN est présentée dans [M.Ester et al, 98] .Il avait prouvé que cet algorithme incrémental donne le même résultat que DBSCAN. Un autre algorithme de clustering GDBSCAN généralisant l'algorithme DBSCAN est présenté dans [Sander et al, 98]. GDBSCAN peut grouper des points selon les deux, leurs attributs numériques et catégoriques. En outre, DBCLASD élimine le besoin de paramètres *MinPts* et *ε* [Xu et al, 98]. OPTICS est présenté dans [Ankerst et al, 99], il généralise DBSCAN en créant un ordre des points qui permet l'extraction de clusters avec des valeurs arbitraires de *ε*.

I.5.4 Clustering basé sur la grille

Les algorithmes basés sur la grille commencent par partitionner l'espace de données à un nombre fini de cellules et accomplir ensuite des opérations exigées sur l'espace partitionné. Les cellules qui contiennent plus qu'un certain nombre de points sont traitées comme denses et les cellules denses sont connectées pour former les clusters [Kotsiantis et al, 01]. La forme de base d'un algorithme basé sur les grilles est la suivante [Steinbach et al, 03]:

1) Diviser l'espace sur lequel les données s'étendent en cellules rectangulaires, par exemple, on divise l'intervalle de valeurs de chaque dimension en cellules de taille égales. La Figure I.14 montre un exemple de grille de deux dimensions.
2) Rejeter les cellules de grille à densité basse. Ceci assume une définition de cluster basée sur la densité, c'est-à-dire, que les régions à haute densité représentent des clusters, alors que les régions à basse densité représentent le bruit. Cette supposition est souvent bonne, bien que les approches basées sur la densité puissent avoir des difficultés quand il y a des clusters de différentes densités.
3) Combiner des cellules adjacentes à haute densité pour former des clusters.

Il y a un certain nombre de problèmes évidents. Les grilles sont carrées ou rectangulaires et elle ne convient pas nécessairement à la forme des clusters. Ceci peut être traité en augmentant le nombre de cellules de la grille, mais cela augmente le temps de calcul. En outre, ces méthodes ne fonctionnent pas bien pour des dimensions moyennes ou élevées.

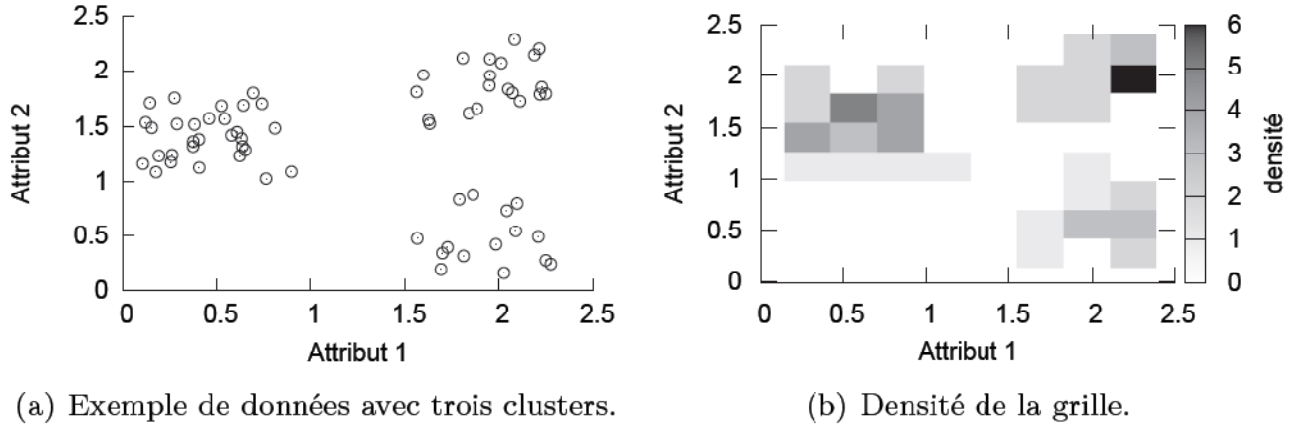


Figure I.15 Un jeu de données à trois clusters et la grille de densité associée [Forestier, 10]

Le processus de clustering dans les méthodes à base de grille consiste à regrouper les cellules denses les plus proches. L'algorithme BANG effectue ce regroupement de manière hiérarchique, en partant de la grille et en fusionnant successivement les cellules denses voisines dont la différence de densité ne dépasse pas un certain seuil.

L'algorithme CLIQUE est une méthode très populaire basée sur les grilles. Il consiste à partir des cellules, et à ne considérer que les cellules dont la densité est supérieure à un seuil. La particularité de CLIQUE est d'explorer plusieurs sous-espaces, c'est-à-dire de considérer plusieurs sous-ensembles des attributs qui décrivent les objets. La grille et les densités sont calculées dans ces sous-espaces, ce qui permet d'effectuer une sélection d'attributs de manière implicite. Ceci permet de ne conserver que les attributs faisant ressortir la densité des cellules, et donc les clusters.

L'utilisation de grilles adaptatives dans ces algorithmes consiste à considérer des grilles non uniformes, c'est-à-dire dont les cellules n'ont pas toutes la même géométrie. En effet, dans le cas où les clusters ont des densités différentes, il peut être intéressant d'avoir des cellules n'ayant pas la même résolution dans tout l'espace des données. Adapter la grille à la densité locale permet d'éviter ce problème. L'algorithme MAFIA propose par exemple une évolution de CLIQUE en créant des grilles adaptatives [Forestier, 10].

I.6 Mesures d'évaluation du clustering

Le clustering vise à regrouper des éléments similaires dans des grappes homogènes, tel que la similarité intra-cluster est maximale tandis que la similarité inter-clusters est minimale. Afin de quantifier ces deux propriétés pour un résultat de clusering donné, deux types d'évaluation peuvent être étudiés [Hatamlou et al, 11].

I.6.1 Mesure d'évaluation externe

Vise à comparer le résultat d'un clustering avec un résultat de référence, fourni généralement par un expert (benchmark). C'est une évaluation entre résultat obtenu et résultat attendu.

I.6.1.1 La pureté

Pour une classe donnée, la pureté représente le pourcentage de données bien classées. Elle se calcule donc à partir de la classe réelle de chaque donnée. En effet pour chacune des classes trouvées nous cherchons la cardinalité du groupe issu de la classe réelle qui est la plus représentée parmi les données de la classe trouvée considérée. Ainsi la somme de toutes les cardinalités pour toutes les classes trouvées moyennée par le nombre total de données représente la valeur de la pureté *PR* (voir équation et l'exemple ci-dessous) [Azzag, 05].

$$P_R = \frac{1}{N} \times \sum_{i=1}^{C_T} max M_i$$

Où *M* est la matrice de confusion (tableau croisé de la dispersion des données entre les classes trouvées et les classes réelles) et N le nombre total de données.

	C_{R1}	C_{R2}	C_{R3}
C_{T1}	75	0	15
C_{T2}	0	30	0
C_{T3}	0	5	80

$$\begin{aligned} P_R &= \frac{\sum_{C_{T1}}^{C_{T3}} = 75 + 30 + 80}{Nombre\ total\ de\ données (= 205)} \\ &= 0,90 \end{aligned}$$

Figure I.16 Exemple d'un calcul de pureté à partir d'une matrice de confusion

La pureté est limité à l'intervalle]0,1] et devrait être maximale.

I.6.1.2 l'erreur couple

Afin de pouvoir évaluer la qualité de la classification obtenue, nous avons utilisé la mesure d'erreur introduite par [Fowlkes et Mallows, 83]. Si, pour chaque donnée di on connaît sa classe réelle CRi et la classe générée par l'algorithme notée CTi, l'erreur de classification *Ec* est calculée comme indiquée dans les équations suivante :

$$Ec = \frac{2}{N(N-1)} \sum_{(i,j)\in\{1,\dots N\}^2, i<j} \epsilon_{ij}$$

N représentant le nombre total de données et :

$$\epsilon_{ij} = \begin{cases} 0\ si\ (C_{Ti} = C_{Tj} \wedge C_{Ri} = C_{Rj}) \vee (C_{Ti} \neq C_{Tj} \wedge C_{Ri} \neq C_{Rj}) \\ 1\ sinon \end{cases}$$

I.6.1.3 F-measure

F-measure combine les idées de la précision et du rappel de la recherche d'information. Elle compare la qualité de clustering en tenant compte des classes correctes connues pour un jeu de données [Rendon et al, 11].

Soit $C = (C_1, C_2, \ldots, C_k)$ des classe générées par l'algorithme et $R = (R_1, R_2, \ldots, R_{k'})$ des classes correctes (du benchmark). Chaque classe R_i contient N_i points de données, chaque cluster C_j est considéré comme l'ensemble de N_j points de données. N_{ij} donne le nombre de points de la classe R_i dans le cluster C_j et N donne le nombre total des points du jeu de données. Pour chaque classe R_i et un cluster C_j, la précision et le rappel sont alors défini comme :

$$\text{Prec}(R_i, C_j) = \frac{N_{ij}}{N_j} \quad et \quad \text{Rep}(R_i, C_j) = \frac{N_{ij}}{N_i}$$

Et la valeur de F-measure correspondante est :

$$Fmes(R_i, C_j) = \frac{2 \cdot \text{Prec}(R_i, C_j) \cdot \text{Rep}(R_i, C_j)}{\text{Prec}(R_i, C_j) + \text{Rep}(R_i, C_j)}$$

F-measure $F(C)$ pour toute la partition est calculée comme :

$$F(C) = \sum_{i=1}^{k'} \frac{N_i}{N} \max_{Cj \in C}(Fmes(R_i, C_j))$$

$F(C)$ est limitée à l'intervalle [0,1], plus $F(C)$ est proche de 1, plus la classification se rapproche de l'optimum.

I.6.2 Mesure d'évaluation interne

Vise à déterminer si le résultat est intrinsèquement cohérent avec les données d'entrée. Cette évaluation n'est donc basée que sur les données et la distance [Rosenberg, 07].

I.6.2.1 L'indice de Dunn

L'indice de Dunn [Dunn, 73] est basé sur l'identification de clusters compacts et bien séparés. Il est défini par le rapport entre la plus petite *dissimilarité interclasse* d_{min} (*i.e.* entre deux individus de deux classes différentes) et la plus grande *dissimilarité intra-classe* s_{max} (*i.e.* entre deux individus de la même classe).

$$Dunn(P) = \frac{d_{min}}{s_{max}}$$

L'objectif principal de cet indice est de maximiser la *dissimilarité interclasse* et de minimiser la *dissimilarité intra-classe*. L'objectif est donc de maximiser l'indice de *Dunn*.

I.6.2.2 L'indice de Dunn généralisé

L'indice de *Dunn généralisé* [Bezdek and Pal, 98] a été introduit après qu'il ait été démontré que l'indice de *Dunn* [Dunn, 73] était non robuste : en effet il dépend uniquement d'un nombre réduit d'individus de la population, et des liens établis entre eux. Ainsi, il est sensible à tout changement qui intervient dans la structure des clusters ainsi qu'aux points aberrants. Les modifications apportées à cet indice interviennent dans le calcul de la dissimilarité *interclasse* et *intra-classe*.

L'indice de *Dunn généralisé* est reconnu comme l'un des indices les plus appropriés pour l'évaluation de la qualité d'une partition donnée, car il fournit un bon compromis entre la *maximisation* de la *dissimilarité interclasse* et la *minimisation* de la *dissimilarité intra-classe* de la partition.

$$Dunn\_gen(P) = \frac{\min\limits_{i=1,\dots,K}\left(\min\limits_{j=1,\dots,K, i\neq j}\left(d_a(C_i, C_j)\right)\right)}{\max\limits_{i=1,\dots,K} s_a(C_i)}$$

La partition *P* produisant la plus grande valeur de *Dunn_gen(P)* correspondra à la meilleure classification.

I.6.2.3 L'indice de Davies-Bouldin

L'indice de *Davies-Bouldin* est une fonction basée sur la minimisation du rapport des dispersions intra-clusters et de la séparation inter-clusters [Davies et al, 79]. Cet indice vise à identifier un ensemble de clusters qui sont compacts et bien séparés [Rendon et al, 11], cet indice est donné par la formule suivante :

$$BD = \frac{1}{c}\sum_{i=1}^{c} Max_{i\neq j}\left\{\frac{S(C_i)+S(C_j)}{d(c_i, c_j)}\right\}$$

Où **c** est le nombre total de clusters générés.

Pour la classe C_i la distance intra classe correspondante est donnée par la formule :

$$S(C_i) = \frac{1}{|C_i|}\sum_{x\in C_i}\left\{\|x - Z_i\|\right\}$$

Où $|C_i|$ est le nombre d'éléments dans la classe C_i, et Z_i est le centroïde de la classe C_i.

La mesure de distance inter clustres C_i et C_j est la distance entre leurs centroïdes Z_i et Z_j respectivement :

$$d(c_i, c_j) = \|Z_i - Z_j\|$$

On constate ainsi que le ratio sera d'autant plus faible que les classes seront compactes et éloignées les unes des autres. Par conséquence, la partition de meilleure qualité sera celle qui minimisera l'indice de *Davies-Bouldin*.

I.6.2.4 L'indice de Silhouette

L'indice de silhouette est défini par [Rousseeuw, 87] pour tout individu X_i de l'ensemble X par la formule suivante :

$$\forall\ X_i \in X;\ \ s(X_i) = \frac{b(X_i) - a(X_i)}{\max\big(a(X_i), b(X_i)\big)}$$

- a(X_i) est la dissimilarité moyenne entre l'individu X_i et tous les autres individus de la classe à laquelle il appartient $C(X_i)$.

$$\forall\ X_i \in X;\ \ a(X_i) = \frac{1}{|C(X_i)| - 1} \sum_{\substack{X_j \in C(X_i) \\ X_j \neq X_i}} d\big(X_i, X_j\big)$$

- b(X_i) est le minimum des dissimilarités moyennes entre l'individu X_i et tous les autres individus des classes de la partition ***P*** différente de $C(X_i)$.

$$\forall\ X_i \in X;\ \ b(X_i) = \min_{\substack{C \in P \\ C \neq C(X_i)}} d(X_i, C)$$

$$\text{où } d(X_i, C) = \frac{1}{|C|} \sum_{X_j \in C} d\big(X_i, X_j\big)$$

On notera que l'indice de silhouette est borné : $-1 \leq s(X_i) \leq 1$. De plus, lorsque $s(X_i)$ est proche de 1, X_i est dit *bien classé* dans $C(X_i)$. Quant $s(X_i)$ est proche de 0, alors X_i se situe entre deux classes.

Finalement, si $s(X_i)$ est proche de -1, X_i est dit *mal classé* dans $C(X_i)$ et doit être rattaché à un autre cluster le plus proche.

Chaque classe est aussi représentée par une silhouette qui montre quels objets sont correctement classés à l'intérieur de cette classe et lesquels n'ont simplement qu'une position intermédiaire. Pour une classe C_i donnée, son indice de silhouette est défini par la moyenne des indices de silhouette des individus qui lui appartiennent :

$$\forall\ C_i \in P;\ s(C_i) = \frac{\sum_{X_j \in C_i} s\big(X_j\big)}{|C_i|}$$

L'indice de silhouette global de la partition ***P*** est donné par la moyenne globale des largeurs de silhouettes dans les différentes classes C_i qui composent la partition :

$$s(P) = \frac{\sum_{C_i \in P} s(C_i)}{k}$$

La meilleure partition retenue est alors celle qui permet d'obtenir une silhouette globale maximale.

I.7 Conclusion

Le problème de clustering a été identifié comme une des problématiques majeures en data mining. Le but de clustering est d'identifier et d'extraire des groupes significatifs dans les données. Pour ce faire, plusieurs méthodes existent dans différentes catégories (c'est-à-dire hiérarchique, par partitionnement, à base de densité et à base de grille).Puisque le clustering est une méthode non-supervisée et il n'y a pas d'indices à priori sur les classes obtenues, on a besoin de certains critères de validation des résultats. Ainsi, tous ces critères de validation nous permettent de justifier les résultats de la classification et d'assurer une bonne compréhension de la structure sous-jacente des données. La suite de la présentation va exposer les approches inspirées de la nature pour le clustering de données, et présenter les applications effectuées sur différents jeux de données.

CHAPITRE II: CLUSTERING PAR DES APPROCHES INSPIRÉES DE LA NATURE

II.1 Introduction

La nature est une immense source d'inspiration pour résoudre des problèmes complexes dans le monde d'informatique, puisqu'elle présente des phénomènes extrêmement divers, dynamiques, robustes, complexes. Elle trouve toujours la solution optimale pour résoudre son problème, tout en maintenant un équilibre parfait entre ses composants. C'est l'initiative derrière l'informatique bio inspirée. Les algorithmes inspirés de la nature sont des méta-heuristiques qui imitent la nature, pour résoudre des problèmes d'optimisation. Durant les dernières décennies, de nombreux efforts de recherche ont été concentrés dans ce domaine. Étant un domaine encore récent, les résultats sont très étonnants, ce qui élargit la portée et la viabilité des algorithmes bio-inspirés en explorant de nouveaux domaines d'application [Binitha et al, 12].

Le problème de clustering de données est identifié comme une des problématiques majeures en extraction des connaissances à partir de données. Le problème de clustering peut être modélisé comme un problème d'optimisation où l'espace de recherche grandit exponentiellement et ne peut pas être parcouru exhaustivement même pour des problèmes de taille moyenne. En effet, le problème de clustering est connu pour être NP-difficile [Bilmes, 97]. Une multitude de méthodes de résolution fait appel à des principes méta-heuristiques. Parmi celles-ci, il existe une branche qui s'inspire de la biologie, c'est les méta-heuristiques d'optimisation inspirées de la biologie. [Bilmes, 97].

II.2 Méta-heuristiques pour l'optimisation difficile

La majorité des problèmes d'extraction de connaissances peuvent s'exprimer comme des problèmes d'optimisation combinatoire. Or, de nombreux problèmes d'optimisation combinatoire sont NP-difficiles et ne pourront donc pas être résolus de manière exacte dans un temps « raisonnable » puisque la capacité de calcul des machines évolue linéairement alors que le temps nécessaire à la résolution de ces problèmes évolue exponentiellement. Lorsqu'on s'attaque à des problèmes réels, il faut se résoudre à un compromis entre la qualité des solutions obtenues et le temps de calcul utilisé.

Au milieu des années 1970 sont apparues des méthodes qui supervisent l'évolution de solutions fournies par des heuristiques. Ces méthodes assurent un compromis entre diversification (quand il est possible de déterminer que la recherche se concentre sur de mauvaises zones de l'espace de recherche) et intensification (on recherche les meilleures solutions dans la région de l'espace de

recherche en cours d'analyse). Ces algorithmes ont été appelés « méta-heuristiques » et ont pour objectif de trouver des solutions dont la qualité est au-delà de ce qu'il aurait été possible de réaliser avec une simple heuristique.

II.2.1 Problème d'optimisation

Un problème d'optimisation au sens général est défini par un ensemble de solutions possibles ***S***, dont la qualité peut être décrite par une fonction objectif ***f***. On cherche alors à trouver la solution ***s**** possédant la meilleure qualité f(***s****) (par la suite, on peut chercher à minimiser ou à maximiser ***f(s)***). Un problème d'optimisation peut présenter des contraintes d'égalité (ou d'inégalité) sur ***s***, être dynamique si ***f(s)*** change avec le temps ou encore multi-objectif si plusieurs fonctions objectifs doivent être optimisées.

Il existe des méthodes déterministes (dites « exactes ») permettant de résoudre certains problèmes en un temps fini. Ces méthodes nécessitent généralement un certain nombre de caractéristiques de la fonction objectif, comme la stricte convexité, la continuité ou encore la dérivabilité. On peut citer comme exemple de méthode la programmation linéaire, quadratique ou dynamique, la méthode du gradient, la méthode de Newton, etc.

II.2.2 Optimisation difficile

Certains problèmes d'optimisation demeurent cependant hors de portée des méthodes exactes. Un certain nombre de caractéristiques peuvent être problématiques, comme l'absence de convexité stricte (multi-modalité), l'existence de discontinuités, une fonction non dérivable, présence de bruit, etc. Dans de tels cas, le problème d'optimisation est dit « difficile », car aucune méthode exacte n'est capable de le résoudre exactement en un temps « raisonnable », on devra alors faire appel à des heuristiques permettant une optimisation *approchée*.

L'optimisation difficile peut se découper en deux types de problèmes : les problèmes discrets et les problèmes continus. Le premier cas rassemble les problèmes de type NP-complets, tels que le problème du voyageur de commerce. Un problème « NP » est dit complet s'il est possible de le décrire à l'aide d'un algorithme polynomial sous la forme d'un sous-ensemble d'instances. Concrètement, il est « facile » de décrire une solution à un tel problème, mais le nombre de solutions nécessaires à la résolution croît de manière exponentielle avec la taille de l'instance. Jusqu'à présent, la conjecture postulant que les problèmes NP-complets ne sont pas solubles en un temps polynomial n'a été ni démontrée, ni révoquée. Aucun algorithme polynomial de résolution n'a cependant été trouvé pour de

tels problèmes. L'utilisation d'algorithmes d'optimisation permettant de trouver une solution approchée en un temps raisonnable est donc courante [Dréo, 04].

Dans la seconde catégorie, les variables du problème d'optimisation sont continues. C'est le cas par exemple des problèmes d'identifications, où l'on cherche à minimiser l'erreur entre le modèle d'un système et des observations expérimentales. Ce type de problème est moins formalisé que le précédent, mais un certains nombre de difficultés sont bien connues, comme l'existence de nombreuses variables présentant des corrélations non identifiées, la présence de bruit ou plus généralement une fonction objectif accessible par simulation uniquement [Dréo, 04].

II.2.3 Heuristiques

Une heuristique d'optimisation est une méthode approchée se voulant simple, rapide et adaptée à un problème donné. Sa capacité à optimiser un problème avec un minimum d'informations est contrebalancée par le fait qu'elle n'offre aucune garantie quant à l'optimalité de la meilleure solution trouvée. Du point de vue de la recherche opérationnelle, ce défaut n'est pas toujours un problème, tout spécialement quand seule une approximation de la solution optimale est recherchée.

II.2.4 Méta-heuristiques

Parmi les heuristiques, certaines sont adaptables à un grand nombre de problèmes différents sans changements majeurs dans l'algorithme, on parle alors de méta-heuristiques. La plupart des heuristiques et des méta-heuristiques utilisent des processus aléatoires comme moyens de récolter de l'information et de faire face à des problèmes comme l'explosion combinatoire. En plus de cette base stochastique, les méta-heuristiques sont généralement itératives, c'est-à-dire qu'un même schéma de recherche est appliqué plusieurs fois au cours de l'optimisation, et directes, c'est-à-dire qu'elles n'utilisent pas l'information du gradient de la fonction objectif. Elles tirent en particulier leur intérêt de leur capacité à éviter les optima locaux, soit en acceptant une dégradation de la fonction objectif au cours de leur progression, soit en utilisant une population de points comme méthode de recherche (se démarquant ainsi des heuristiques de descente locale). Souvent inspirées d'analogies avec la réalité (physique, biologie, éthologie, . . .), elles sont généralement conçues au départ pour des problèmes discrets, mais peuvent faire l'objet d'adaptations pour des problèmes continus.

II.3 Méta-heuristiques biomimétiques pour le Clustering

Nous nous intéressons dans cette partie aux algorithmes de clustering qui utilisent des méta-heuristiques d'optimisation inspirées de la biologie et aux avantages qu'ils apportent pour le problème de clustering. Nous allons donner un aperçu de ces algorithmes. Nous ne traiterons pas ici les

approches neuronales mais plutôt les approches à base de population d'agents (algorithmes évolutionnaires, fourmis artificielles, intelligence en essaim, systèmes immunitaires) puisqu'elles constituent le contexte de notre travail.

II.3.1 Clustering par algorithmes évolutionnaires

Les algorithmes évolutionnaires sont des techniques de recherche inspirées par l'évolution biologique des espèces. Parmi plusieurs approches, les algorithmes génétiques en constituent certainement l'exemple le plus connu. Le principe d'un algorithme évolutionnaire est décrit dans la figure II.1. Un ensemble de *N* points dans un espace de recherche, choisis a priori au hasard, constituent la population initiale, chaque individu *x* de la population possède une certaine performance, qui mesure son degré d'adaptation à l'objectif visé : dans le cas de la minimisation d'une fonction objectif *f*, *x* est d'autant plus performant que *f(x)* est plus petit. Un AE consiste à faire évoluer progressivement, par générations successives, la composition de la population, en maintenant sa taille constante.

Au cours des générations, l'objectif est d'améliorer globalement la performance des individus, on s'efforce d'obtenir un tel résultat en mimant les deux principaux mécanismes qui régissent l'évolution des êtres vivants, selon la théorie de C. Darwin [Dréo et al, 03]:

- La *sélection*, qui favorise la reproduction et la survie des individus les plus performants,
- Et la *reproduction*, qui permet le brassage, la recombinaison et les variations des caractères héréditaires des parents, pour former des descendants aux potentialités nouvelles.

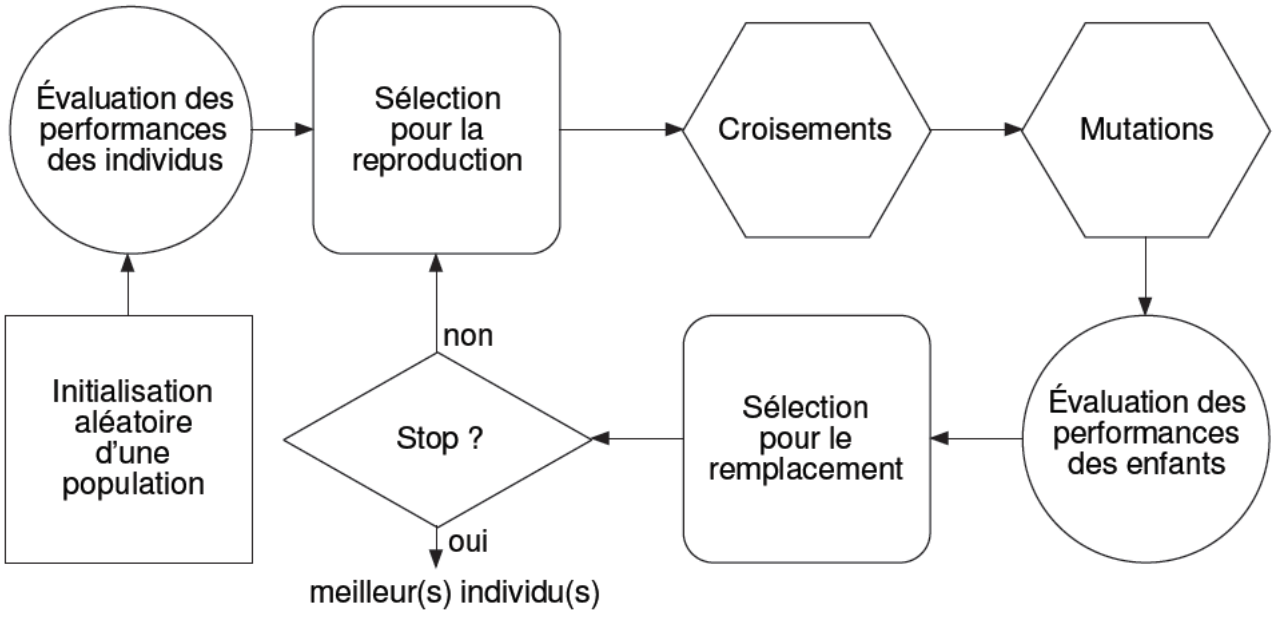


Figure II.1 Principe d'un algorithme évolutionnaire standard [Dréo et al, 03]

Ce cycle est répété jusqu'à satisfaction d'un critère d'arrêt, c'est-à-dire soit après que *n* générations aient été créées, soit après la génération d'une solution suffisamment satisfaisante au sens de la fonction d'évaluation.

[Greene, 03] propose une méthode permettant de constituer des hiérarchies de partitions. Il décrit tout d'abord une nouvelle méthode descendante, subdivisant récursivement une population de départ en sous-populations, en essayant d'optimiser une mesure tenant compte de l'inertie intra-groupes et intergroupes et de la taille des groupes constitués. Cette méthode est dépendante des conditions initiales et en particulier de l'ordre dans lequel sont insérés les objets dans l'arbre. Greene propose alors de générer le meilleur arbre possible par application d'un algorithme génétique. Pour cela, une population initiale d'arbres est générée en choisissant un ordre aléatoire d'insertion des objets. Une mesure de la qualité de chaque arbre est effectuée, puis ceux-ci sont sélectionnés par roulette proportionnelle élitiste (les deux meilleures solutions sont conservées). Le croisement de deux arbres consiste à choisir les meilleures branches du premier niveau de chaque arbre et à les réunir. Ce croisement peut produire des solutions non-valides de deux façons. Un objet peut se retrouver dans deux classes simultanément, il est alors supprimé de la moins bonne. Un objet peut être absent de la partition, il est alors réinséré par l'algorithme original. L'opération de mutation consiste à supprimer un objet aléatoirement de l'arbre et à le réinsérer dans l'arbre par l'algorithme original, ce qui peut changer la constitution des groupes.

Cette méthode, qui a pour originalité d'être hiérarchique et de générer un nombre de classes *k* libre, ne fournit pas d'indication sur l'optimalité de la solution générée.

II.3.2 Clustering par fourmis artificielles

Les sociétés d'insectes telles que les fourmis, constituent aussi des sources d'inspiration pour le développement de systèmes artificiels « Les sociétés d'insectes nous proposent un modèle de fonctionnement bien différent du modèle humain: un modèle décentralisé, fondé sur la coopération d'unités autonomes au comportement relativement simple et probabiliste, qui sont distribuées dans l'environnement et ne disposent que d'informations locales. » [Deneubourg, 91]. Toutes ces propriétés peuvent être mises à profit pour la conception d'algorithmes de résolution de problèmes.

L'exemple des fourmis est le plus répandu dans la littérature à cause de leur capacité à réaliser des tâches hautement complexes à partir des interactions simples avec une intelligence très rudimentaire [Dorigo, 96]. En particulier deux comportements collectifs ont été principalement étudiés chez les fourmis : l'optimisation de chemin et le tri des éléments du couvain.

Le premier comportement met en évidence la capacité des fourmis à optimiser la procédure de fourragement. En effet, au départ les fourmis se déplacent d'une manière aléatoire de la fourmilière vers une source de nourriture en laissant des traces de phéromones sur le chemin qu'elles empruntent, cette trace tend à attirer les congénères. Il a été constaté alors que les fourmis qui choisissent l'itinéraire le plus court, le terminent le plus vite. Les fourmis vont alors préférer progressivement cet itinéraire car il est marqué avec une plus grande quantité de phéromones [Goss, 89]. De plus l'évaporation naturelle de la phéromone renforce encore plus ce choix des fourmis par l'affaiblissement des chemins les plus longs.

Le second comportement collectif des fourmis concerne leur aptitude à nettoyer leur nid en organisant collectivement des cimetières composés de cadavres empilés les uns sur les autres. Le principe est le suivant : plus un cadavre est isolé, plus la fourmi a de chances de ramasser ce cadavre. La probabilité pour une fourmi porteuse de déposer ce qu'elle transporte suit une règle inverse : plus le monticule observé est important, plus la probabilité de déposer le corps au sol sera grande. Les chercheurs ont exploité ce comportement pour fournir des algorithmes de classification pour lequel l'informatique classique n'a pas donné de solution satisfaisante.

Les algorithmes de classification automatique sont un autre type d'algorithmes inspirés de comportements collectifs observés chez les fourmis : le tri collectif de couvains ou la constitution de cimetières. Les premiers travaux dans ce domaine ont été ceux de Deneubourg et son équipe [Deneubourg, 90], se basant sur une colonie de fourmis artificielles qui se déplacent aléatoirement sur une grille rectangulaire et sont capables de ramasser et de déposer des objets présents sur une grille dans le but de les regrouper selon un critère de similarité. Ces travaux ont été par la suite améliorés et étendus à différents domaines d'application.

Deneubourg était l'origine des algorithmes de classification par les fourmis. Lors des expériences de simulation, les objets à classer sont placés aléatoirement sur une grille à deux dimensions. Les fourmis sont placées aussi aléatoirement sur la grille. Chaque fourmi n'a qu'une perception locale de son environnement et a pour tâche de déplacer les objets en fonction de la densité des objets de même type dans leur environnement proche appelé « voisinage ».

Le principe est de regrouper les objets similaires en des groupes sur une grille. Chaque fourmi peut prendre un objet avec une probabilité en fonction de sa similarité avec les objets présents dans son voisinage et le déposer selon la même probabilité. Après un certain nombre d'itérations, des groupes d'objets similaires se forment sur la grille. La principale caractéristique de ces algorithmes est

leur coté non supervisé qui permet de découvrir automatiquement le nombre de groupe adéquat sans intervention extérieure comme dans les algorithmes classiques de classification. Les opérations de ramassage et de dépôt des objets sont représentées par les probabilités P_p et P_d :

$$P_p = (k1/k1+f)^2 \qquad P_d = (f/k2+f)^2$$

Où f est une estimation du nombre d'objets placés dans le voisinage de la fourmi (la proportion d'éléments perçus dans le voisinage de la fourmi). k1 et k2 sont des constantes positives. Quand il y a peu d'objets dans le voisinage de l'objet convoité par la fourmi, f << k1 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 >> k1 et alors p_p est proche de 0. Deneubourg utilise k1 = 0.1 et de k2 = 0.3 [Deneubourg, 90].

Comme on le constate le mécanisme de rétroaction positive est présent mais agit d'une manière différente par rapport à celui décrit dans les algorithmes d'ACO. Les fourmis ne communiquent plus par trace de phéromones mais c'est la distribution des objets sur la grille qui est à la base de leur communication et coopération indirecte « stigmergie »

L'algorithme proposé par Deneubourg a été repris et étendu par Lumer et Faita [Lumer, 94] pour la classification des données numériques. Les extensions introduites concernent en particulier les points suivants :

- Les données sont représentées par des vecteurs de caractéristiques (numériques).
- La similarité entre deux données est mesurée comme une distance euclidienne entre leurs vecteurs de caractéristiques.
- La fourmi est capable de percevoir une région R_S de SxS cases autour de sa position courante sur la grille.

La Figure II.2 représente un exemple de grille utilisé dans l'algorithme de Faieta et Lumer. Les objets sont représentés par des cases de deux motifs décrivant leurs types et le rectangle en trait épais est la région perçue par la fourmi.

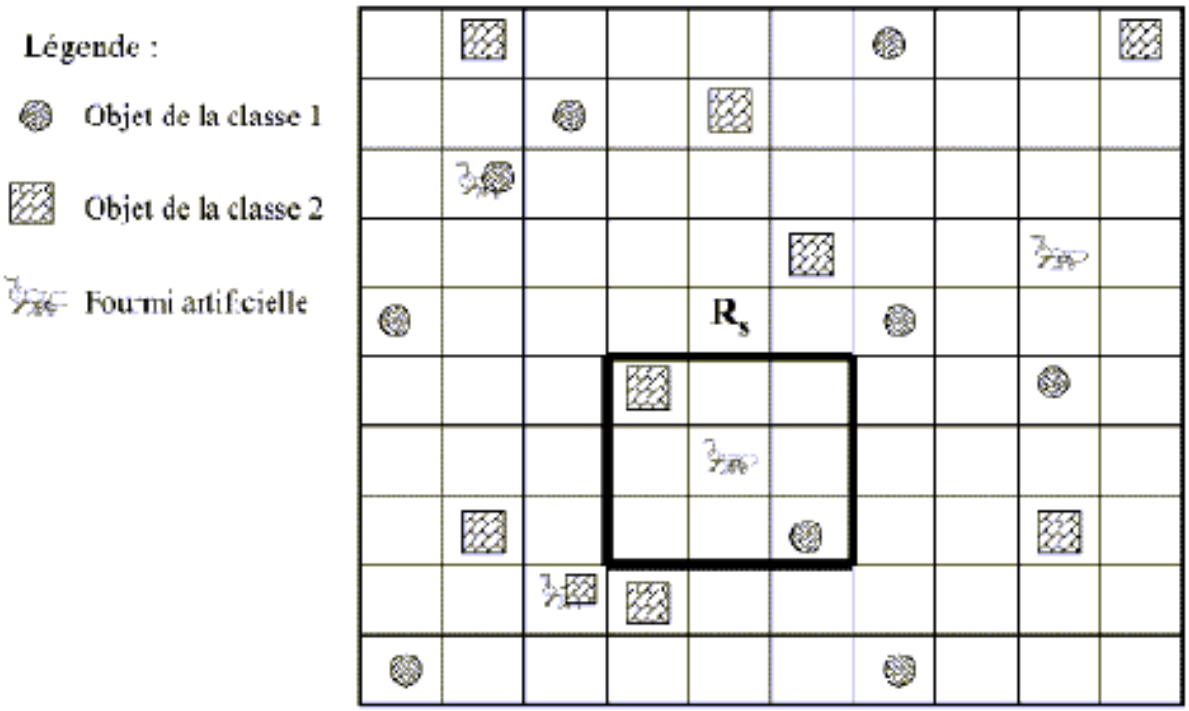


Figure II.2 Grille de classification de Lumer et Faieta [Lumer, 94]

Les probabilités de déplacement et de dépôts des objets deviennent alors [Lumer, 94] :

$$P_p(i)=\left(\frac{k_1}{k_1+f(i)}\right)^2$$

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

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

r(i) est la position de l'objet *i* sur la grille. *f(i)* est une mesure de similarité moyenne de l'objet *i* avec les objets *j* de son entourage. Le facteur *α* contrôle la consistance (stabilité) de la fonction de dissimilarité entre les objets. Si *α* est trop élevé les objets différents seront mis dans la même classe dans le cas contraire les objets similaires ne seront pas regroupés ensembles. Les tests ont été menés avec k1=0.1, k2=0.15, R=9, α= 0.5

Voici ci-dessous l'algorithme de Lumer et Faita. *A* fourmis sont utilisées pour effectuer la tâche de classification de *N* objets.

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

Figure II.3 l'algorithme de Lumer et Faita [Lumer, 94]

En se basant sur les travaux de Lumer et Faieta, Monmarché proposa un nouvel algorithme de classification AntClass [Monmarché, 99]. AntClass utilise une colonie de fourmis artificielles qui se déplacent sur une grille toroïdale carrée afin d'éviter les effets de bords. La taille de la grille est calculée automatiquement en fonction de la taille des objets à classer. Chaque fourmi a la capacité de transporter plusieurs objets à la fois et de placer un tas d'objets sur une seule case de la grille. La localisation et l'extraction des classes est plus facile et plus fiable que dans le modèle de Lumer et Faieta. De plus, AntClass est une hybridation de l'algorithme stochastique de fourmis et de l'algorithme déterministe de classification « le K-means » afin d'accélérer la convergence vers une partition stable. AntClass comporte deux exécutions successives d'un motif constitué d'une application de l'algorithme de fourmis suivi de l'algorithme K-means. Les fourmis génèrent une partition initiale de bonne qualité qui sera par la suite raffinée par K-means. AntClass a été testé sur des bases de données réelles et les résultats obtenus sont significatifs.

Dans [Kanade, 03] Kanade et Hall reprennent les travaux de Monmarché et combinent l'algorithme de fourmis qu'il a proposé avec l'algorithme classique de classification FCM [Bezdeck, 81]. Dans une première étape, l'algorithme de fourmis est utilisé initialement pour fournir une première classification qui sera par la suite raffinée en utilisant l'algorithme FCM. Dans une seconde étape, les centres de classes obtenus par FCM sont considérés comme de nouveaux objets à classer. Ces derniers

seront par la suite déplacés et fusionnés par les fourmis. A la fin, les classes obtenues par l'algorithme de fourmis sont une autre fois raffinée par l'algorithme FCM.

Ramos et al [Ramos, 02] ont étendus les travaux de Deneubourg et al [Deneubourg, 91] et Lumer and Faieta's [Lumer, 94] en proposant leur algorithme Acluster pour la classification de mots dans des documents. ACluster utilise les traces de phéromones comme moyen de communication entre les fourmis et comme mémoire commune à toutes les fourmis du système de classification. Si une classe disparaît de la grille, la phéromone tend à s'évaporer dans sa direction, ce qui permet d'éviter que les fourmis se dirigent vers des directions non intéressantes. Ce travail a été repris par la suite par Abraham et Ramos [Abraham, 03] et appliqué dans le domaine de l'internet. Dans le même ordre d'idée, Handel [Handl, 03] propose une variante de l'algorithme de Lumer et Faita pour la recherche de documents pour les moteurs de recherche sur internet. Les modifications introduites concernent en particulier l'adaptation de la valeur du paramètre α en fonction du taux de ramassage et de dépôt d'objets par les fourmis et l'utilisation d'un processus de diversification nommé « stagnation control » afin d'accélérer la convergence. De plus les classes d'objets sont obtenues en appliquant un algorithme hiérarchique.

Labroche dans sa thèse a introduit un nouveau modèle à base de fourmis pour la classification utilisant le système d'identification chimique des fourmis [Labroche, 02]. Dans la nature, les éthologistes ont montré que chaque fourmi arrive à reconnaître ses congénères qui appartiennent à sa colonie à partir d'une odeur coloniale qui est le fruit des apports génétiques, environnementaux et comportementaux. A partir de ce mécanisme d'identification un nouvel algorithme de classification a été proposé dans lequel chaque donnée est une fourmi dont l'odeur est déterminée par les valeurs prises par les attributs décrivant cette donnée. Les fourmis effectuent des rencontres aléatoires et décident qu'elles appartiennent ou non à la même classe en fonction de cette odeur.

En s'inspirant toujours du comportement des fourmis, Azzag proposa dans [Azzag, 04] un nouvel algorithme de classification automatique qui trouve son origine dans la manière dont les fourmis réelles forment des structures vivantes. Il s'agit d'une méthode de classification hiérarchique distribuée qui simule le phénomène d'auto-assemblage observé chez les fourmis pour regrouper les données selon un arbre. Chaque fourmi représente une donnée à classer. A partir d'un point de support fixe sur lequel sont situées initialement les fourmis (répartition des données), 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 (grande similarité avec les fourmis de la structure).

II.3.3 Clustering par essaim de particules

La méta-heuristique basée sur la méthode des essaims particulaires ("Particle Swarm Optimization", PSO) a été développée par Kennedy et al. en 1995 [Kennedy et al, 95]. Le principe de la méthode provient analogiquement avec les comportements collectifs d'animaux comme le déplacement des bancs de poissons ou le vol des oiseaux. En effet, on peut observer chez ces animaux des dynamiques de déplacement relativement complexes, alors qu'individuellement chaque individu a une intelligence limitée et une connaissance seulement locale de sa situation dans l'essaim.

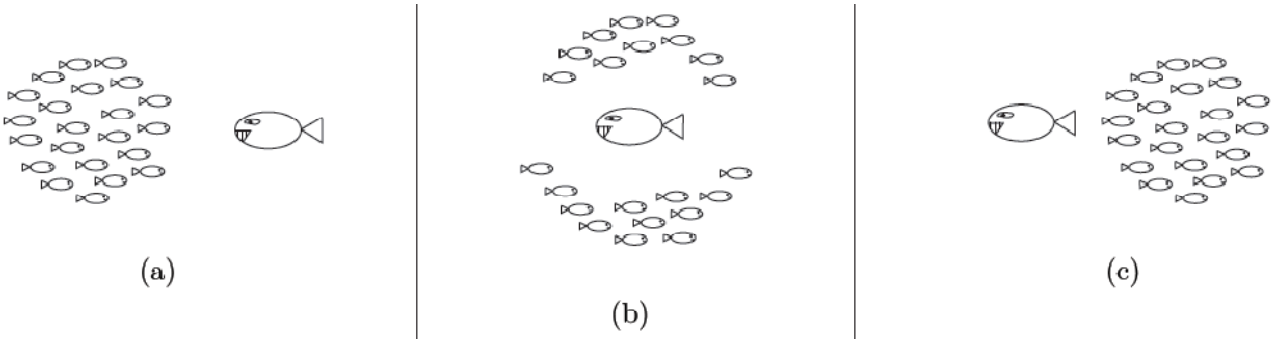


Figure II.4 Auto-organisation après perturbation par un prédateur [Dréo, 04]

Les individus de l'algorithme sont appelés particules et la population est appelée essaim. Un individu de l'essaim n'a pour connaissance que la position et la vitesse de ses plus proches voisins. Chaque individu utilise donc, non seulement, sa propre mémoire, mais aussi l'information locale sur ses plus proches voisins pour décider de son propre déplacement. Des règles simples, telles que, aller à la même vitesse que les autres, se déplacer dans la même direction, ou encore rester proche de ses voisins : sont des exemples de comportements qui suffisent à maintenir la cohésion de l'essaim, et qui permettent la mise en œuvre de comportements collectifs complexes et adaptatifs. L'intelligence globale de l'essaim est donc la conséquence directe des interactions locales entre les différentes particules de l'essaim.

Les particules formant l'essaim ont un comportement relativement simple. Cependant ces comportements individuels peuvent faire émerger un comportement complexe sans aucun contrôle central grâce à des interactions élémentaires entre particules, ou entre particules et environnement. L'intelligence en essaim tente alors de simuler les mécanismes produisant ce type de comportements, afin de proposer de nouvelles techniques pour la résolution de problèmes.

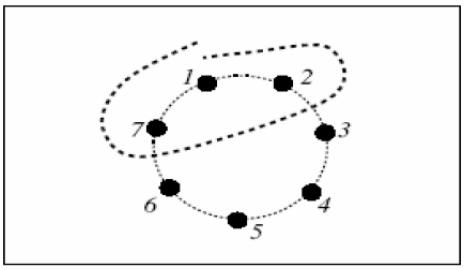


Figure II.5 Le cercle virtuel pour un essaim de sept particules [clerc, 03]

Le groupe d'information de taille trois de la particule 1 est composé des particules 1, 2 et 7. Au départ de l'algorithme, un essaim est réparti au hasard dans l'espace de recherche, chaque particule ayant également une vitesse aléatoire. Ensuite, à chaque pas de temps :

- Chaque particule est capable d'évaluer la qualité de sa position et de garder en mémoire sa meilleure performance, c'est-à-dire la meilleure position qu'elle a atteinte jusqu'ici (qui peut en fait être parfois la position courante) et sa qualité (la valeur en cette position de la fonction à optimiser).
- Chaque particule est capable d'interroger un certain nombre de ses congénères de son voisinage et d'obtenir de chacune d'entre elles sa propre meilleure performance.
- A chaque pas de temps, chaque particule choisit la meilleure des meilleures performances dont elle à connaissance, modifie sa vitesse en fonction de cette information et de ses propres données et se déplace en conséquence.

A partir des quelques informations dont elle dispose, une particule doit décider de son prochain mouvement, c'est-à-dire décider de sa nouvelle vitesse. Pour ce faire, elle combine trois informations :

- Sa vitesse actuelle.
- Sa meilleure position actuelle.
- La meilleure performance (vitesse et position) de ses voisines.

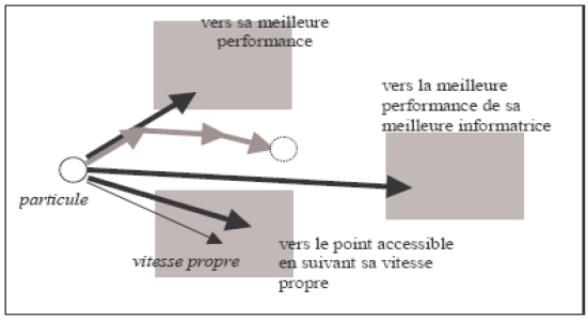


Figure II.6 Schéma vectoriel du déplacement d'une particule [Ou et Lin, 06]

En 1998, ces principes ont été appliqués pour la première fois à un problème de classification de données [Proctor et Winter, 98]. Les agents représentent chacun une donnée. Un agent réagit aux autres agents présents dans son voisinage en tenant compte de la similarité des données. Un agent se déplacera plutôt vers des données qui lui sont similaires. Cette règle comportementale permet donc de former des groupes de données similaires.

Dans [Picarougne et al, 05], cet algorithme a été amélioré et évalué d'une manière plus systématique. Une distance idéale entre individus est définie, distance qui dépend de la similarité entre les données. Un critère d'arrêt est utilisé également en mesurant l'entropie spatiale du nuage d'agents.

Merwe et Engelbrecht ont proposé une approche basée sur les essaims de particules [Merwe et al, 03]. Une particule représente un vecteur de K centroïdes de clusters, où K est le nombre de clusters et il doit être spécifié par l'utilisateur. Par conséquent, un essaim représente un certain nombre de vecteurs candidats pour un jeu de données. L'algorithme de PSO commence avec une population de vecteurs choisis aléatoirement, ensuite pour chaque vecteur, on assigne les points de données au centroïde de cluster le plus proche en terme de distance Euclidienne. La fitness de chaque vecteur est mesurée par l'erreur de quantification. L'évolution de la population est accomplie en ajustant les centroïdes de chaque vecteur par apport au meilleur vecteur de centroïde trouvé et le meilleur vecteur dans le voisinage de ce vecteur.

Dans [Xiao et al, 03], les auteurs utilisent une méthode hybride basée sur les essaims de particules et les cartes organisatrices (Self-Organizing Maps) pour réaliser un clustering des gènes dans des expérimentations d'expression génique. Les algorithmes à essaim de particules ont été utilisés aussi pour le clustering des images, l'algorithme cherche les centroids d'un nombre de clusters spécifié par l'utilisateur, chaque cluster regroupe les primitives d'images similaires [Omran, 05].

II.3.4 Clustering par système immunitaire artificiel

Le système immunitaire est responsable de la défense de l'organisme contre les maladies et autres infections. Il distingue les éléments qui appartiennent à l'organisme (le soi) de ceux qui sont nuisibles à l'organisme « antigène » (le non soi), pour cela plusieurs cellules contribuent pour l'élimination des antigènes, ces cellules « B » participent pour ce qu'ont appelle « la réponse immunitaire biologique ». Nous distinguons deux types de réponse immunitaire, une réponse immunitaire innée, qui représente les défenses présentes dès la naissance de l'individu, et une

réponse immunitaire adaptative, c'est l'ensemble des défenses apprises ou acquises au cours du temps [Emi, 06].

La réponse immunitaire innée a rarement intéressée les informaticiens, à cause de ses capacités statiques, réagissant à l'infection sans pour autant apprendre à y réponde de manière plus efficace. Par contre, la réponse immunitaire adaptative est d'un grand intérêt dans le domaine informatique, principalement grâce à ses capacités d'apprentissage adaptatif. L'immunité spécifique contient deux mécanismes principaux, l'un est responsable de la distinction entre le soi et le non soi, l'autre permet au système de mémoriser les antigènes rencontrés. Lorsqu'un nouvel antigène pénètre dans le corps, la réponse immunitaire passe par les étapes suivantes (principes de la sélection clonale [Roitt, 90]):

- Au début, la concentration de l'antigène est tellement faible que seule l'immunité innée est activée. Comme l'antigène est nouveau, aucune cellule B n'est assez spécifique pour se lier avec ;
- Au fur et à mesure que l'antigène se développe, sa concentration devient assez élevée pour activer les cellules B les moins spécifiques ;
- Une fois les cellules B activées, elles vont se multiplier pour produire un grand nombre de clones. Chaque clone est une cellule B identique à la cellule qui la produite. Le nombre de clones est proportionnel à l'affinité de la liaison cellule B – antigène ;
- Pour augmenter la spécificité des anticorps et l'efficacité de la réponse immunitaire, les clones entrent dans une phase d'hyper mutations, modifiant ainsi la structure de leurs récepteurs (anticorps). comme les mutations sont aléatoires, les cellules obtenues (dites matures) peuvent devenir plus spécifiques ou moins spécifiques ;
- Lorsque la concentration de l'antigène diminue (à cause de la réponse immunitaire), seules les cellules B les plus spécifiques continuent à être activées, les autres (les moins spécifiques) ne sont plus activées et finissent par mourir. Ceci a pour effet de rendre la population de cellules B de plus en plus spécifique à chaque génération ;
- Après maturation, les cellules B deviennent soit des cellules plasma, soit des cellules mémoire. Les cellules plasma sont de véritables usines à anticorps capable d'en produire en quantités impressionnantes. Les cellules mémoire quand à elles, vont survivre longtemps après la disparition de l'antigène, et peuvent une fois activée produire de grandes quantités d'anticorps en très peu de temps.

Lors de la réponse primaire (un nouvel antigène pénètre dans le corps), il faut plusieurs jours avant que des anticorps efficaces apparaissent dans le sang. Mais lors de la réponse secondaire (un antigène similaire pénètre dans le corps) et grâce aux cellules mémoire issues de la première infection, les anticorps correspondants sont produits beaucoup plus rapidement et en plus grandes quantités. On dit que le système est devenu immunisé contre l'antigène, ou encore qu'il l'a mémorisé.

En ce qui concerne la classification, les principes des systèmes immunitaires est le suivant (voir par exemple le système aiNet [Castro et al, 00]) : les données *d1, ..., dn* représentent les antigènes. Ces antigènes sont présentés itérativement au système jusqu'à l'obtention d'une condition d'arrêt. On suppose que les données sont numériques, et donc qu'un antigène est un vecteur de dimension n. A chaque itération, l'antigène présenté va activer des anticorps (assimilés dans cette modélisation à des lymphocytes-B). Un anticorps est également représenté par un vecteur de dimension n. Les anticorps suffisamment proches de l'antigène (au sens de la distance euclidienne - L'affinité -) vont subir des clonages avec mutation (interaction anticorps/antigènes) afin d'amplifier et d'affiner la réponse du système. Également, ces anticorps vont subir une sélection (interaction anticorps/anticorps) : ceux qui sont trop proches les uns des autres seront diminués en nombre. Après ces itérations, le système converge en plaçant des anticorps (qui agissent comme des détecteurs) de manière judicieuse et en nombre adapté aux données.

II.4 Conclusion

Le problème de clustering peut être modélisé comme un problème d'optimisation où l'espace de recherche grandit exponentiellement et ne peut pas être parcouru exhaustivement même pour des problèmes de taille moyenne. Le problème de clustering est connu pour être NP-difficile et l'utilisation des méta-heuristiques d'optimisation inspirées de la biologie offre beaucoup d'avantages à ce problème.

Les techniques de fouille de données présentées ci-dessus représentent une partie des techniques existantes pour l'extraction des données. Ils peuvent prédire les futures tendances et actions à partir de l'historique des données, permettant ainsi de prendre les bonnes décisions. Ce grand nombre de techniques est dû qu'ils n'ont pas tous le même objet, et qu'aucune n'est optimale dans tous les cas et qu'elles s'avèrent en pratique complémentaires les unes des autres en les combinant intelligemment.

CHAPITRE III: CONTRÔLE D'ÉMERGENCE

III.1 Introduction

L'approche par émergence, appelée aussi la résolution des problèmes par émergence a prouvé son efficacité pour les problèmes dits complexes tant pour le domaine de l'intelligence artificielle que pour le domaine d'optimisation. Le contrôle est décentralisé et la totalité des individus participent ensemble au processus de résolution du problème.

La théorie de la science des systèmes complexes cherche à comprendre, modéliser et simuler les comportements manifestés au niveau global du système, en se basant sur les deux propriétés : l'auto-organisation et l'émergence où une fonction de niveau supérieur peut émerger à travers les éléments de niveau inférieur [Forbes, 04].

Les fourmis sont des sources fascinantes d'inspiration pour les systèmes informatiques car elles sont auto-organisées et exhibent des comportements émergents, et démontrent de l'intelligence collective. C'est la raison derrière notre choix dans cette thèse.

III.2 Le phénomène d'émergence

III.2.1 Définition

L'émergence signifie qu'il y a apparition au niveau macro d'un phénomène nouveau non observé au niveau micro. Il est alors impossible de décrire, de prédire ou d'expliquer à base des conditions de base définies au niveau micro. Le système doit être capable de modifier son comportement en fonction de son environnement sans l'aide d'un superviseur [Gleize, 04].

III.2.2 Les propriétés de l'émergence

Comme rapporté dans [Georgé, 04] et [Gleize, 04], on trouve dans la littérature plusieurs particularités attribuées à l'émergence, à savoir :

- Nouveauté et imprévisibilité : l'émergence présuppose qu'il y a apparition de nouvelles propriétés, structures, formes ou fonctions, et d'autre part, elle implique qu'il est impossible de décrire, d'expliquer ou de prédire ces nouveaux phénomènes en terme physique à partir des conditions de bases définies aux niveaux inférieurs.
- Temporalité : le phénomène doit se construire dans le temps, il ne peut être instantané.
- Cohérence et corrélation : le phénomène a une identité propre mais liée d'une manière ou d'une autre aux parties de micro-niveau.

- Irréductibilité : Ceci implique l'impossibilité de déduire, à partir des propriétés du micro-niveau, les propriétés du macro-niveau. On ne peut pas réduire le phénomène à la somme de ses parties.
- Interdépendance des niveaux : Langton dans [Langton, 90], définit l'émergence en termes de relation de feed-back (rétroactions) entre les niveaux dans un système dynamique. Les micro-dynamiques locales causent les macro-dynamiques globales, et les macro-dynamiques globales contraignent les micro-dynamiques locales.

III.3 L'auto-organisation comme technique d'émergence

III.3.1 Définition

L'auto-organisation est un processus dans lequel un modèle de niveau global, émerge d'un grand nombre de composantes en interactions. De plus, les règles spécifiant les interactions entre composantes de système utilisent uniquement des informations locales, sans référence à aucun modèle global [Camazine, 01].

III.3.2 Les propriétés de l'auto-organisation

Nous présentons les quatre propriétés fondamentales caractérisant l'auto-organisation. Pour lequel l'absence de l'une de ces propriétés, le système est considéré comme non auto-organisé.

- **Contrôle central :** Le système est en interaction continue avec l'environnement, capable de recevoir des stimulus et produire des effets, sans aucun contrôle central, ni un planning donnant au système la manière de réaction.
- **Incrémentation dans l'ordre :** Le système peut avoir une incrémentation dans l'ordre comme étant le résultat d'un processus auto-organisé. Cependant l'ordre ne s'incrémente pas indéfiniment et ne restera pas tout le temps dans une configuration d'ordre supérieur. L'ordre est en alternance entre la croissance et la décroissance.
- **Adaptabilité :** Un système auto-organisé doit être robuste vis-à-vis les perturbations, et capable de s'adapter aux changements survenus dans l'environnement.
- **Interaction :** Sans interaction, les composantes ne sont qu'un groupe indépendant d'entités, incapables d'exprimer des comportements collectifs et cohérents.

Les trois premières propriétés sont un consensus dans la littérature [De Wolf et al, 04]. La quatrième est mentionnée dans [Bonabeau et al, 99] avec le feedback positif et feedback négatif et la fluctuation et ses amplifications. Il est intéressant de mentionner ici que l'émergence est différente de

concept de l'auto-organisation par la présence de nouveaux comportements globaux [De Wolf et al, 04].

III.4 L'intelligence en essaim

Les études éthologistes ont pu identifier chez certains animaux et insectes des comportements auto-organisés menant à l'émergence de tâches qualifiées de complexe sans contrôle global [Bonabeau, 97]. En effet, ces insectes sociaux sont capables de créer des communautés géantes et réalisent collectivement des tâches extraordinaires telles que le fourragement chez les fourmis, la construction de véritables cathédrales de terre chez les termites, les déplacements collectifs chez les oiseaux et les bancs de poissons. Ce comportement global complexe émerge à partir des comportements locaux et simples de chaque élément du groupe [Van Dyke Parunak, 97].

Cette capacité à passer de comportements individuels simples à des comportements collectifs complexes est appelée "intelligence en essaim" ("swarm intelligence") ou « intelligence collective » sous domaine de l'intelligence artificielle distribuée. L'intelligence en essaim tente alors de simuler les mécanismes produisant ce type de comportements, afin de proposer de nouvelles techniques pour la résolution de problèmes.

III.5 Exemples de systèmes artificiels émergents

III.5.1 Automates cellulaires

Les Automates Cellulaires ont été inventés par deux des plus renommés mathématiciens du siècle dernier. Stanislas Ulam et John von Neumann. Ils sont remarquables par leur simplicité. Le « Jeu de le Vie » est une des instances les plus étudiées de ces automates cellulaires [Gardner, 70].

Il suffit de concevoir une grille, comme pour un jeu de Dames, où chaque case peut prendre la valeur 0 ou 1. Le 0 signifiant que la case est « morte », le 1, qu'elle est « vivante ». Ensuite, chaque « cellule » ainsi définie, possède quelques règles très simples de fonctionnement qui régissent son évolution :

- Si une cellule est morte, et elle a exactement 3 voisines vivantes, elle devient vivante.
- Une cellule vivante survit si elle a exactement 2 ou 3 voisines vivantes.
- Dans tous les autres cas, la cellule meurt ou reste morte (i.e. 4 ou plus, elle est étouffée, 1 ou 0. elle meurt d'isolement).

Que peut-on faire avec des règles aussi simples ? En fait, si on pose certaines configurations de cellules sur une grille, on peut observer des phénomènes surprenants. L'exemple le plus populaire reste le « planeur » (Figure III.1) qui montre une évolution du planeur sur 5 étapes. On retrouve la même configuration de cellules à l'étape 5, qu'à l'étape 1, mais elles sont décalées vers le has et vers la droite.

Il existe un grand nombre de ces configurations remarquables. Elles ont toutes en commun le fait d'être régies par les mêmes règles de base et que le résultat observable interprété ne peut être prédit.

Dans une vision plus informatique des choses, la théorie veut que l'on puisse ainsi simuler une Machine de Turing, les signaux transitant sous forme de groupes de cellules comme le planeur, et venant modifier l'état de certaines configurations stables.

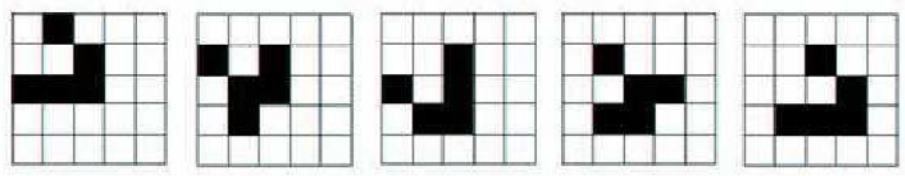


Figure III.1 L'évolution du planeur sur 5 étapes

III.5.2 Algorithmes de colonie de fourmis

Il existe de nombreux phénomènes émergents issus du comportement des insectes sociaux tels que les fourmis ou les termites. Cela n'a rien de surprenant car si l'on prend par exemple une colonie de fourmis, ce système, bien évidement complexe, contient tous les ingrédients pour des phénomènes émergents : grand nombres d'entités autonomes, nombreuses interactions, fonctionnement simple des entités, capacités perceptives et cognitives limitées, système confronté à un environnement dynamique, fonctionnements de haut niveau remarquables et nécessaires à la survie. Les algorithmes de colonies de fourmis sont nés à la suite d'une constatation : les insectes sociaux en général, et les fourmis en particulier, résolvent naturellement des problèmes relativement complexes. Dans cette société d'insectes, on peut observer l'apparition de phénomènes de haut niveau qui ne sont pas décrits dans le fonctionnement des parties.

Parmi les problèmes étudiés par les biologistes, les problèmes de choix lors de l'exploitation de sources de nourriture. Certaines espèces de fourmis ont la particularité d'employer la stigmergie pour communiquer via des substances volatiles appelées phéromones. Elles sont attirées par ces substances

qu'elles perçoivent. Les fourmis peuvent déposer des phéromones au sol, et forment ainsi des pistes odorantes, qui pourront être suivies par leurs congénères.

Les fourmis utilisent les pistes de phéromone pour marquer leur trajet, par exemple entre le nid et une source de nourriture. Au départ, un grand nombre de fourmis se déplacent à l'extérieur du nid, plus ou moins au hasard. En recherchant de la nourriture, elles déposent une légère trace de phérormones tout le long de leur chemin. Si l'une d'entre elles découvre une ressource quelconque, elle retourne au nid en déposant une trace beaucoup plus intense (cette intensité dépend éventuellement de la richesse de la ressource). Cette trace tend à attirer les congénères qui, en la suivant, vont parvenir à la nourriture. Ils vont alors retourner au nid et renforcer la trace à leur tour.

On assiste ainsi à la mise en place d'une boucle de rétroaction positive : la trace attire des individus qui renforcent la trace qui attire donc plus d'individus. Les fourmis tendent à suivre les pistes de phéromones qu'elles rencontrent, mais il ne s'agit que d'une tendance. A tout moment, la probabilité existe qu'un individu quitte la trace puis se déplace plus ou moins au hasard. A cette occasion, la fourmi égarée peut éventuellement découvrir une source de nourriture beaucoup plus riche que celle qu'exploitent ses congénères. En déposant alors une trace de phéromone plus intense encore, elle va les attirer vers cette nouvelle ressource, formant une nouvelle boucle de rétroaction positive. Enfin, quand la satiété produit son effet, ou que la ressource est épuisée, une boucle de rétroaction négative se met en place. Dans ce cas, si l'on considère que les traces de phéromones s'évaporent assez rapidement, une fois que la nourriture sera épuisée, de moins en moins de fourmis auront tendance à suivre la trace qui va finir par disparaitre.

Cet exemple est intéressant par sa simplicité car il permet de bien imaginer le fonctionnement du système et donc de comprendre qu'il est logique que les fourmis finissent par choisir le chemin le plus court. Mais pourtant, lorsque l'on analyse seulement les règles des fourmis, on n'y reconnait pas la recherche du plus court chemin. Ce n'est qu'en considérant le système dans sa globalité que ce phénomène s'explique, Il y bien une différence de niveau micro/macro et on peut parler de phénomène émergent. Les algorithmes inspirés des colonies de fourmis ont été appliques, aussi bien pour, l'optimisation combinatoire [Dorigo, 96], la classification [Chen et al, 04], ou pour le routage de paquets sur les réseaux de communication [Caro et Dorigo, 98].

III.6 Emergence contrôlée de l'essaim

III.6.1 Introduction

Dans certains systèmes naturels et techniques, on peut observer qu'une petite fraction des individus peut influencer un groupe entier et de cette manière avoir un effet sur le comportement émergent au niveau du système. Par exemple considérons le modèle de comportement de déplacement collectif des groupes d'animaux. En ce modèle le mouvement émergent global résulte des règles locales des individus. Les modèles récemment développés de groupes d'animaux en mouvement divisent la population en individus naïfs et individus informés [Couzin et al, 05]. Alors que les individus naïfs suivent les règles classiques de mouvement collectif, les individus informés de la sous-population mettent à jour leurs orientations selon une moyenne pondérée des règles sociales « normales » et une direction « préférée » fixe, partagée par tous les individus au courant. Ces modèles peuvent expliquer comment une petite fraction des individus informés peut guider le groupe entier dans une direction désirée [Janson et al, 05]. Également l'introduction d'individus artificiels peut influencer les directions de déplacement et le comportement d'agrégation des essaims d'animaux réels, comme montré dans des expériences avec les poissons [Sumpter et al, 08]. Une petite fraction des individus (artificiels), en utilisant un comportement légèrement différent, peut significativement influencer le mouvement collectif émergent d'un groupe d'animaux.

Un sous-ensemble des individus peut ainsi contrôler des effets émergents au niveau du système. L'émergence contrôlée de l'essaim veut dire l'introduction des *individus (agents) de contrôle* dans un système pour contrôler des effets émergents au niveau système. Le nouveau système constitué de individus usuels et des individus de contrôle présentent des propriétés nouvelles, c'est à dire, l'effet émergent qui est prévu à être contrôlé peut disparaître ou se changer. Même des nouveaux effets peuvent se produire. De cette façon, « pour contrôler l'émergence » est un effet émergent lui-même, basé sur l'interaction d'un essaim d'individus de contrôle avec le système.

III.6.2 Le contrôle d'une boucle de rétroaction

Différentes catégories des distributions peuvent être assumées pour ces boucles de rétroaction. Dans le cas le plus centralisé, une boucle globale contrôle le système et essaie d'atteindre des objectifs donnés. Ici la partie contrôle a une vue globale sur le système et permet de mesurer les états du système global, si nécessaire. Dans les grands systèmes distribués une telle boucle de rétroaction globale n'est pas toujours applicable, car parfois, il n'est pas possible de collecter toutes les informations nécessaires dans un temps raisonnable, ou la quantité de ces données est trop grande.

Dans tels cas, une organisation distribuée décentralisée des boucles de rétroaction, chaqu'une d'elle contrôle seulement des sous-systèmes [Cakar et al, 07]. Les boucles de rétroaction locales distribuées n'ont qu'une vue locale sur le système et leurs interventions de contrôle n'affectent que les sous-systèmes spécifiques. Il s'agit d'un point important, car la réalisation d'un objectif global pour le système entier doit émerger à travers la réalisation des objectifs locaux des boucles locales de rétroaction introduites. Finalement, la conception des boucles locales de rétroaction décentralisées peut conduire à nouveau au problème de la façon de concevoir des effets émergents.

A côté du problème de boucles de rétroaction dans les grands systèmes décentralisés, il y a des situations où il n'est même pas possible d'ajouter une boucle de rétroaction à un système. Si un système est déjà en cours d'utilisation et la nécessité pour le contrôle n'apparaissait pas dans sa phase de conception, il peut être trop coûteux de modifier le système [Mogul, 06].

III.6.3 Comparaison entre l'émergence contrôlée de l'essaim et la boucle de rétroaction

Une comparaison schématique d'une approche de contrôle basée sur la boucle de rétroaction et l'approche de l'émergence contrôlée de l'essaim est donnée sur la figure III.2. Les cercles et leurs connexions représentent les composants d'auto-organisation du système avec leurs relations et leurs interactions locales. Sur un niveau plus élevé du système émergence se produit. Dans le système de contrôle basé sur la boucle de rétroaction représentée dans la figure III.2 (a), les mesures locales et globales de l'état du système sont observées et sur la base de ces informations et connaissances le système est influencé pour contrôler l'effet émergent. Pour imposer le contrôler, les composants des systèmes sont souvent conçus pour avoir une sorte d'interfaces de contrôle.

En comparant le système contrôlé d'essaim représenté dans la figure III.2 (b) il n'existe pas de type particulier d'effecteurs de contrôle nécessaires. Le système est contrôlé par l'introduction de composants de contrôle (représentés par des cercles avec un «C») qui infèrent de la même manière avec d'autres composants, comme le font les composants habituels. Une condition préalable pour appliquer un contrôle d'essaim est que le système doit être évolutif en termes de composants, c'est à dire, il doit y avoir la possibilité d'introduire et d'intégrer de nouveaux composants dans le système, ou au moins il doit y avoir la possibilité de substituer une partie des composants par des composants de contrôle. La généralité de la méthode ne devrait pas être surestimée et l'applicabilité doit être considérée d'un système à l'autre.

Figure III.2 (a) Le contrôle de l'émergence par l'introduction d'une boucle de rétroaction suivant l'approche centralisée observateur/contrôleur.

Figure III.2 (b) Le contrôle de l'émergence en utilisant l'approche émergence contrôlée de l'essaim par l'introduction d'éléments de contrôle.

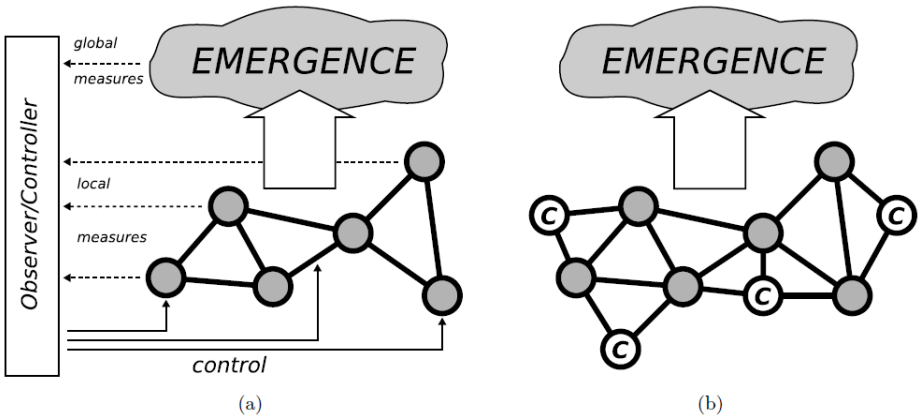


Figure III.2 Comparaison entre l'émergence contrôlée de l'essaim et la boucle de rétroaction [Scheidler, 10]

Pour concevoir le comportement des composants de contrôle entraînent le problème de la façon de concevoir des effets émergents en général. A côté de l'utilisation de «modèles de conception pour des effets émergents», par exemple pris des systèmes auto-organisés naturels, il a été suggéré qu'un processus manuel ou automatique par essais et erreurs peut aider à trouver des règles locales appropriées. Cela signifie que dans le cas de la conception de composants de contrôle, des expériences doivent être faites en appliquant les implémentations de test des composants de contrôle pour les systèmes réels ou en les testant au moyen de simulations.

L'introduction d'un deuxième essaim des composants avec un comportement différent dans un système n'est pas seulement utile en termes de contrôle des effets émergents, mais elle peut aussi conduire à une meilleure compréhension des effets émergents dans le système considéré. Par exemple, elle peut montrer la robustesse d'un effet émergent, c.-à-d., combien de composants de "mauvais comportement" le système peut gérer.

III.7 L'émergence contrôlée de l'essaim dans le clustering basé sur les fourmis

III.7.1 Un modèle de la formation de pile par les fourmis

Le comportement collectif des fourmis concerne leur aptitude à nettoyer leur nid en organisant collectivement des cimetières composés de cadavres empilés les uns sur les autres. (Comme détaillé dans la section II.3.2). Les opérations de ramassage et de dépôt des objets sont représentées par les probabilités:

$$p_{pick}^{clust}(f) = \left(\frac{k^+}{k^+ + f}\right)^2 \qquad \text{and} \qquad p_{drop}^{clust}(f) = \left(\frac{f}{k^- + f}\right)^2$$

[Scheidler, 10] dans sa thèse a introduit le concept d'anti-clustering, ce concept est détaillé dans la section suivante.

III.7.2 Anti-Clustering

L'objectif est de construire des agents de contrôle qui se comportent similaire aux agents de clustering standard, mais qui sont capable, lorsqu'ils sont ajoutés à un système de clustering existant, de réduire (ou d'empêcher) l'effet de clustering. Nous appelons ces agents de contrôle: les agents anti-clustering, ou les agents-AC. Comme mentionné précédemment, les différents types d'agents anti-clustering, qui seront introduites dans la suite, se comportent de la même manière que les agents de clustering. Les agents de clustering et les agents-AC ne diffèrent que dans les distributions de probabilité pour déposer et ramasser les objets.

III.7.2.1 Agents-AC renversés

Pour les agents-AC renversés, les probabilités pour qu'un agent de clustering ramasse un objet ou le dépose, sont permutées dans une certaine situation. Un tel agent va déposer (respectivement ramasser) un objet avec la même probabilité qu'un agent de clustering ramasserait (respectivement déposerait) un objet dans le même voisinage *f* :

$$p_{pick}^{rev}(f) = p_{drop}^{clust}(f) = \left(\frac{f}{r^+ + f}\right)^2$$

$$p_{drop}^{rev}(f) = p_{pick}^{clust}(f) = \left(\frac{r^-}{r^- + f}\right)^2$$

Où r+ et r− sont les valeurs seuils, utilisées pour les agents renversés. Dans la plupart des expériences les deux paramètres r = r+ = r− sont initialisés à une même valeur.

Comme un cas extrême du comportement de l'agent-AC renversé, nous introduisons les agents-AC renversés stricts, ou tout simplement les agents-AC stricts. Ils ramassent les objets avec une probabilité 1 s'il y a au moins un autre objet dans leur voisinage et ne déposent les objets que s'il n'y a aucun élément dans le voisinage. Un tel comportement est dénoté par r = 0, puisque c'est la même chose que le comportement des agents-AC renversés pour les paramètres seuil r+ = r− proches de zéro.

Une autre possibilité de revenir sur le comportement des agents de clustering est de renverser la probabilité de ramasser et de déposer les objets, de sorte que:

$$p_{pick}^{rev}(f) = 1 - p_{pick}^{clust}(f) \text{ and } p_{drop}^{rev}(f) = 1 - p_{drop}^{clust}(f)$$

III.7.2.2 Agents-AC de voisinage inversé

Agents-AC de voisinage inversé ont les mêmes règles comportementales comme les agents de clustering, mais ils ont une perception inversée du voisinage, c'est à dire, sur chaque cellule voisine, un agent-AC de voisinage inversé voit un objet lorsque la cellule est vide, sinon il ne voit aucun objet. Ainsi, ces agents déposent des objets avec une probabilité plus élevée lorsqu'il y a moins d'objets dans leur voisinage. Formellement, la probabilité de ramasser un objet et de le déposer :

$$p_{pick}^{inv}(f) = p_{pick}^{clust}(1-f) = \left(\frac{i^+}{i^+ + 1 - f}\right)^2$$

$$p_{drop}^{inv}(f) = p_{drop}^{clust}(1-f) = \left(\frac{1-f}{i^- + 1 - f}\right)^2$$

Où i+ et i- sont des valeurs seuils utilisées pour les agents-AC de voisinage inversé.

III.7.2.3 Agents-AC aléatoires

L'introduction de l'aspect aléatoire dans le comportement des agents, dans le sens que des objets sont transportés aux cellules aléatoires, ça peut évidemment gêner un clustering robuste. Les agents-AC aléatoires toujours ramassent des objets quand ils entrent dans une cellule occupée. Si un tel agent porte un objet, il le dépose sur une cellule vide avec une probabilité fixe t > 0. Formellement, pour les agents-AC aléatoires, la probabilité de ramasser un objet et de le déposer :

$$p_{pick}^{rand}(f) = 1 \qquad p_{drop}^{rand}(f) = t$$

Dans la figure III.3 des exemples de fonctions de probabilité pour ramasser ou déposer des objets pour les différents agents sont donnés [Scheidler, 10].

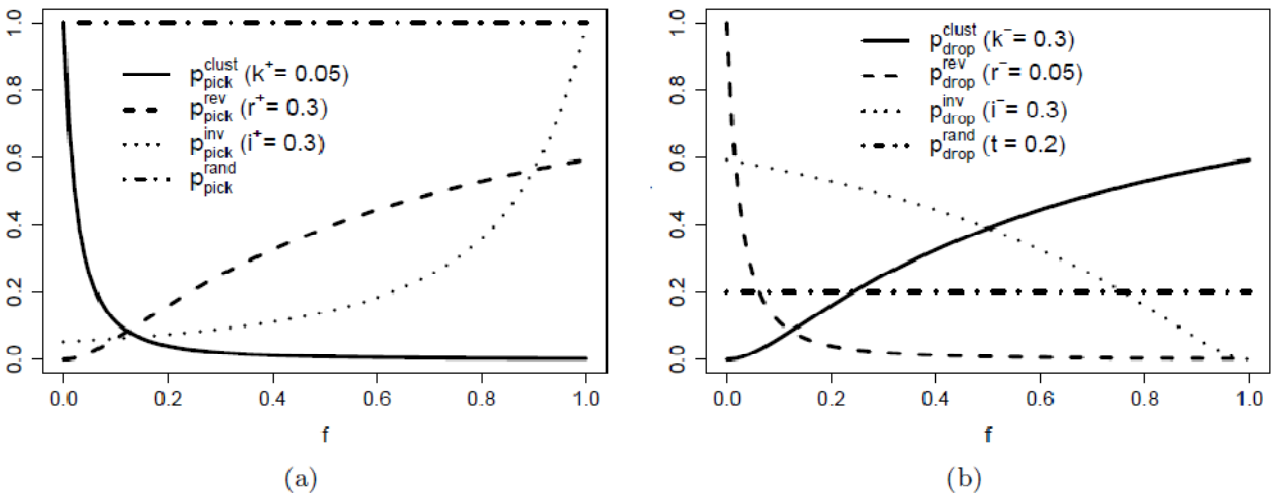


Figure III.3 Les différents types d'agents [Scheidler, 10]

Probabilités pour ramasser (a) et de déposer (b) un objet pour les différents types d'agents (agents de clustering, agents-AC renversés, agents-AC de voisinage inversé et agents-AC aléatoires); f dénote la fraction des cellules occupées dans le voisinage.

III.8 Conclusion

Les systèmes complexes à organisation autonome se composent de nombreux composants autonomes. Les effets émergeants peuvent se produire sans qu'ils soient ni voulus, ni prévus dans la phase de conception. Pour rendre les systèmes fiables, il est nécessaire de prendre soin de ce problème. Le contrôle de l'émergence consiste à introduire des boucles de rétroaction pour rendre les systèmes auto-adaptatifs.

Une autre approche a été discutée pour contrôler ou empêcher des effets émergents (indésirable), appelée émergence contrôlée de l'essaim. Cette approche introduit un essaim d'agents de contrôle supplémentaires dans le système. Cette approche a été testée pour empêcher le clustering émergent dans le modèle de classification chez les fourmis. Trois différents types d'agents de contrôle ont été étudiés pour ce système à savoir, les agents-AC renversés, les agents-AC de voisinage inversé et les agents-AC aléatoires.

CHAPITRE IV: CONTRIBUTION

IV.1 Introduction

Dans les chapitres précédents nous avons vu en détail tout ce qui se rapporte au problème du clustering et les différentes approches proposées dans la littérature. Cette étude bibliographique souligne le fait que, bien que des travaux relativement récents soient rapportés dans la littérature, aucune méthode ne s'impose comme meilleure que les autres à ce jour.

L'approche que nous proposons est introduite dans ce chapitre. Dans cette approche, nous proposons d'utiliser une méta-heuristique biomimétique s'inspirant du comportement des fourmis réelles, à des fins de clustering, tout en essayant d'exploiter le comportement collectif des fourmis avec contrôle de l'émergence issue de ce comportement. Ce qui distingue notre approche par rapport à ce qui existe dans la littérature est qu'elle arrive à se résoudre à un compromis entre le temps d'exécution et la qualité des solutions obtenues.

IV.2 Formulation du problème de clustering

Étant donné un ensemble d'objets S qu'on veut regrouper en k clusters. Formellement, il s'agit de trouver la partition $C^* = \{c_1^*, c_2^*, \dots, c_k^*\}$ de S où chaque c_i représente un cluster qui optimise une mesure de qualité. La partition doit satisfaire les conditions suivantes:

1. $\forall i \quad c_i \neq \phi$
2. $\forall i, j \quad c_i \cap c_j = \phi$
3. $\bigcup_i c_i = S$

Le nombre de partitions qu'il est possible de former croit de manière considérable en fonction de la cardinalité de S et du nombre de clusters. En effet, le nombre de façons de regrouper n objets dans k clusters est approximatif à $k^n/k!$. Résoudre un tel problème par une méthode exacte peut s'avérer impossible. C'est la raison pour laquelle il est plus judicieux d'utiliser une métaheuristique. Dans le cadre de notre travail, on propose de traiter le problème sous l'angle de l'optimisation distribuée en considérant une méta-heuristique basée population plus précisément l'optimisation par les colonies de fourmis artificielles avec une mise en œuvre sur une plateforme multi-agents à savoir NETLOGO.

IV.3 Idées de base de l'approche proposée

Notre approche se base sur trois idées essentielles, qui sont :

- Une nouvelle stratégie adoptée par la fourmi, où chaque fourmi utilise une courte mémoire locale dans laquelle elle mémorise chaque objet rencontré dans son parcours, la taille de cette mémoire est fixe (une liste gérée selon la politique First In First Out). Contrairement aux méthodes classiques, où la fourmi ne perçoit que son voisinage.

- La modification précédente exige une modification au niveau des opérations de ramassage et de dépôt des objets (pick-up & drop).

- L'introduction d'agents anti-clustering en réponse à l'émergence négative. Ces agents agissent de façon inverse par rapport au comportement de fourmis artificielles.

Afin de tirer profit du parallélisme intrinsèque des colonies de fourmis et faciliter la mise en œuvre du mécanisme de contrôle d'émergence nous avons opté pour l'utilisation de l'environnement NETLOGO (voir *annexe-B* pour plus de détails sur NETLOGO).

IV.4 Représentation de données et de solutions

La classification des données numériques via l'utilisation de l'ACO exige une représentation des données qui s'adapte au comportement de la fourmi artificielle comme suit :

- Les données sont représentées par des vecteurs de caractéristiques (numériques).
- La similarité entre deux données est mesurée comme une distance euclidienne entre leurs vecteurs de caractéristiques.

Chaque vecteur de caractéristiques est représenté dans une case sur une grille à 2 dimensions.

IV.5 Approche proposée

L'approche proposée est basée sur trois idées principales qui sont :

IV.5.1 Nouvelle stratégie adoptée par la fourmi

La taille de la grille est calculée automatiquement en fonction de la taille des objets à classer. Chaque fourmi a la capacité de transporter un objet à la fois et de le placer sur une seule case de la grille. Les objets à classer sont placés aléatoirement sur une grille à deux dimensions. Les fourmis sont placées aussi aléatoirement sur la grille. Chaque fourmi n'a qu'une perception locale de l'objet rencontré, elle empile les caractéristiques de l'objet rencontré dans sa liste (ses attributs), si la liste atteint la taille maximale (huit cases), une case est dépilée et les caractéristiques de l'objet courant seront empilées selon la politique FIFO. Plus un objet est isolé, plus la fourmi a de chances de le

ramasser. La probabilité pour une fourmi porteuse de déposer ce qu'elle transporte suit une règle inverse, plus le monticule observé est important, plus la probabilité de déposer l'objet sera grande.

Le principe est de regrouper les objets similaires en des groupes sur une grille. Chaque fourmi peut prendre un objet avec une probabilité en fonction de sa similarité avec les objets présents dans son chemin parcouru et le déposer selon une autre probabilité. Après un certain nombre d'itérations, des groupes d'objets similaires se forment sur la grille. Les opérations de ramassage et de dépôt des objets sont représentées par les probabilités P_{pick} et P_{drop} :

$$P_{pick} = (k * 100 / (k * 100 + f))^2 \qquad P_{drop} = (k / (k + (1 - f)))^2$$

Où f est une estimation du nombre d'objets placés dans la liste de la fourmi qui sont jugés similaires (jugé similaire si la distance <= un seuil). k est une constante fixée à 0.01 dans notre approche.

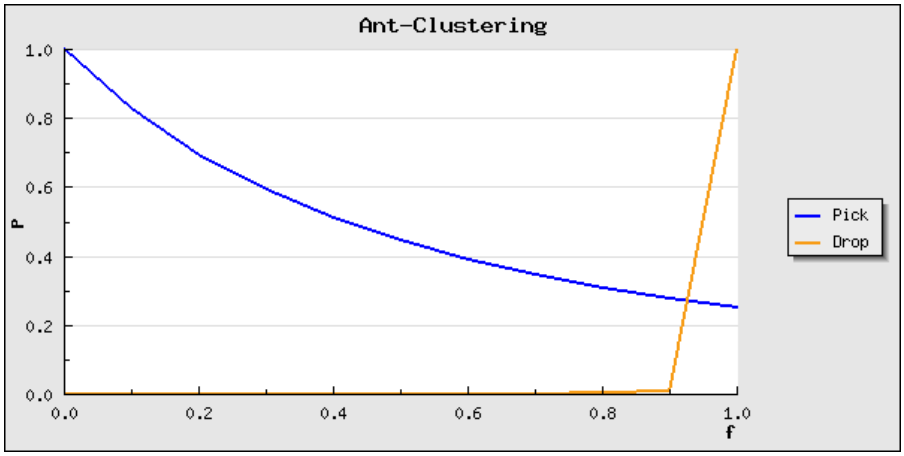


Figure IV.1 Probabilités de Pick & Drop de notre approche en fonction de f

Si la valeur de f est petite, c.à.d. quand il y a peu d'objets qui sont jugés similaires dans la liste, ceci implique que P_{pick} est égale ou proche de 1 et l'objet a beaucoup de chance d'être ramassé, et vice-versa. Si la valeur de f est suffisamment élevée, c.à.d. quand tous les objets sont jugés similaires dans la liste (f = 1), ceci implique que P_{drop} est égale à 1 et l'objet est forcement déposé, et vice-versa.

IV.5.2 Modification de Pick & Drop

Tous les travaux mentionnés dans le chapitre-II utilisaient les opérations de ramassage et de dépôt des objets P_{pick} et P_{drop} :

$$P_{pick} = (k1/k1+f)^2 \qquad P_{drop} = (f/k2+f)^2$$

Les valeurs de k1 et de k2 sont respectivement initialisées à 0.1 et 0.3 (d'autres travaux initialisaient k1 à 0.01)

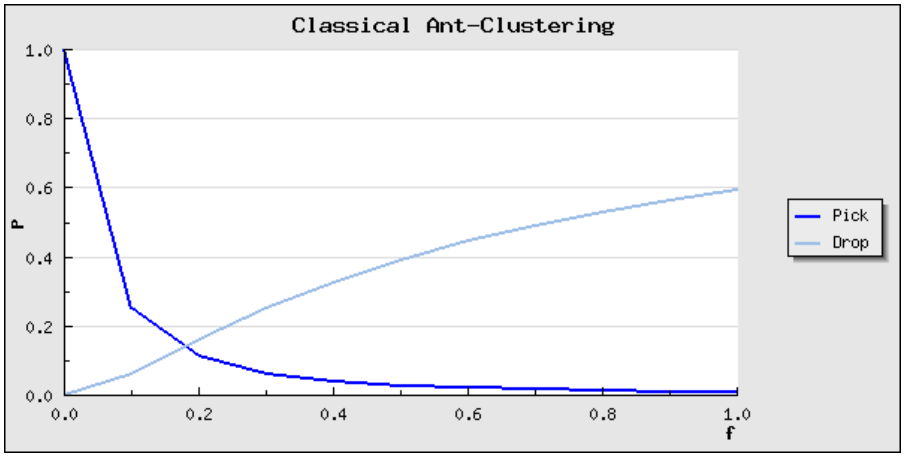


Figure IV.2 Probabilités de Pick & Drop classiques en fonction de f

La courbe de drop dans la figure IV.2 montre bien que, quelque soit la valeur de f, implique que $(f/k2+f)^2 < 0.6$; et comme l'opération drop n'a lieu qu'après la vérification de cette inéquation : random-float 1.0 < $(f/k2+f)^2$, ce qui montre que, malgré tous les objets sont jugés similaire (f = 1), il y a une chance de moins de 60% pour que l'objet détenu soit déposé ! (si random-float 1.0 < 0.6).

La courbe de pick dans la figure IV.2 montre bien que, les objets isolés sont surement ramassés, $(k1/k1+f)^2 = 1$ si f = 0 ; mais les objets qui sont jugés moins isolés ont une petite chance à être ramassés, car l'opération pick n'a lieu qu'après la vérification de cette inéquation : random-float 1.0 < $(k1/k1+f)^2$.

Dans notre approche on veut que l'opération de drop n'ait lieu que s'il y a un taux de similarité f égale à 1, c'est-à-dire la formule classique de drop ne s'accorde pas avec notre besoin, et c'est la raison pour laquelle on l'a modifiée. La courbe de drop dans la figure IV.1 montre bien la différence.

De même, la formule classique de pick aussi ne s'accorde pas avec notre besoin, car on veut prioritairement ramasser les objets isolés, et ramasser aussi les objets qui sont jugés moins isolés (pour des raisons de qualité de clusters, ça s'est avéré d'après les expériences). La courbe de pick dans la figure IV.1 montre bien la différence.

IV.5.3 Introduction d'agents anti-clustering

Les agents anti-clustering ont été utilisés par [Scheidler, 10], dont l'objectif est de construire des agents de contrôle qui se comportent contrairement aux agents de clustering, afin de réduire (ou d'empêcher) l'effet de clustering.

Dans notre approche l'utilisation des agents anti-clustering est pour la destruction de l'émergence négative, c'est-à-dire la destruction des clusters non appropriés, c'est-à-dire si le nombre

de clusters générés ne s'assortit pas avec le nombre voulu. Cette intervention n'est que pour donner la chance aux agents de clustering pour reconstruire de nouveaux clusters sur une plateforme initiale.

Les opérations de ramassage et de dépôt des objets que les agents anti-clustering utilisent sont représentées par les probabilités $P_{anti\text{-}pick}$ et $P_{anti\text{-}drop}$:

$P_{anti\text{-}pick} = (k * 100 / (k * 100 + (1 - f)))^2$ $\quad$ $P_{anti\text{-}drop} = (k / (k + f))^2$

Où f est une estimation du nombre d'objets placés dans la liste de la fourmi qui sont jugés similaires. k est une constante positive fixée à 0.01 dans notre approche.

On voit bien que la courbe de pick dans la figure IV.1 est strictement inversée par rapport à la courbe de pick dans la figure IV.3. Même chose pour les courbes de drop dans les deux figures.

La figue IV.3 montre bien que l'opération de drop n'ait lieu que s'il y a un taux de similarité f inférieur ou égale à 0.1, c'est-à-dire le dépôt de l'objet détenu se fait dans une zone jugée dissimilaire par rapport à cet objet.

De même, la courbe de pick de la figure IV.3 montre bien qu'on veut prioritairement ramasser les objets qui sont complètement similaires (f=1), et ramasser aussi les objets qui sont jugés moins isolés.

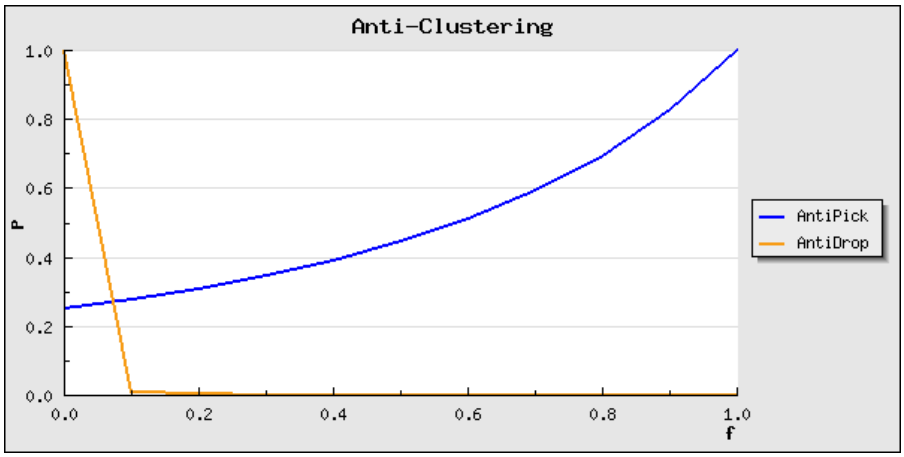


Figure IV.3 Probabilités de Pick & Drop par un agent anti-clustering en fonction de f

IV.6 Approche proposée : description

La démarche de cette approche suit trois étapes principales. La section suivante détaille chacune d'elles.

IV.6.1 Seuil de similarité

Cette approche essentiellement s'appuie sur la mesure du seuil de similarité, initialement la variable threshold s'initialise à une valeur comprise entre 0 et MaxDist, où MaxDist est la distance la plus longue entre deux objets. L'algorithme s'exécute avec cette valeur de ce threshold, si le nombre de clusters générés est supérieur ou inférieur à celui voulu (émergence négative), une modification sera apportée à la valeur de threshold tout en utilisant la méthode dichotomie, c'est-à-dire on donne à l'algorithme une nouvelle tentative de clustering. Avant de passer à exécuter l'algorithme avec cette nouvelle valeur de threshold (nouvelle tentative), il faut détruire les clusters générés, car ils sont considérés comme une émergence négative.

Il s'agit d'un contrôle d'une boucle de rétroaction, cette boucle globale contrôle le système et essaie d'atteindre des objectifs donnés (un nombre de clusters générés égale à celui voulu). Ici la partie contrôle a une vue globale sur le système et permet de mesurer les états du système global. Cette mesure s'effectue à chaque nombre limité d'itérations.

IV.6.2 Clustering initial des données

L'application des notions préalablement décrites engendre des clusters de tailles différentes, on peut distinguer deux types de clusters générés en fonction de leurs tailles : des clusters suffisamment mûrs qui ont dépassé l'étape préliminaire, et des clusters ayants un nombre réduit d'objets qu'on l'appelle outliers (inférieur ou égale à quatre). Les clusters mûrs seront les noyaux des clusters outliers. Cette procédure est décrite en détails dans la section suivante.

IV.6.3 Fusionner les ouliers avec les clusters générés

Cette procédure consiste à affecter chaque objet issu d'un cluster outliers à un cluster mûr censé le plus proche par rapport à cet objet. Le nombre de clusters mûrs sera le nombre final de clusters générés, et la fusion des outliers n'est que pour accélérer le processus de clustering, car l'approche s'appuie sur le contrôle itératif de chaque tentative de clustering, ce qui nous impose à réduire le temps d'exécution de chaque tentative, qui va à son tour réduire le temps total d'exécution. Car à l'origine, on voudrait se résoudre à un compromis entre ce temps et la qualité de clustering.

IV.7 Algorithme proposé pour le custering des données avec contrôle d'émergence

L'algorithme suivant récapitule notre approche. *A* fourmis sont utilisées pour effectuer la tâche de classification de *N* objets, et MaxDist est la distance la plus longue entre deux objets (la section précédente est la description de cet algorithme).

Soit la variable threshold recevant (MaxDist / 2)

Début : Placer aléatoirement les N objets o_1,…….o_N sur la grille

- *Pour T=1 à T_{max} faire*
  - *Pour tout aj ∈ {a_1,…….a_A} faire*
    - *Déplacer la fourmi sur une case voisine*
    - *Si la fourmi tombe sur un objet alors*
      - *La fourmi l'empile dans sa liste*
      - *Si la taille de la liste atteint 9 alors la fourmi dépile un objet selon la politique FIFO*
    - *FinSi*
    - *Si la fourmi a_j ne transporte pas d'objets et elle tombe sur un objet o_i alors*
      - *Calculer $f(o_i)$ et $P_{pick}(o_i)$*
      - *Si $P_{pick}(o_i)$ > random-float 1.0 alors la fourmi aj ramasse l'objet oi*
    - *Sinon*
      - *Si la fourmi a_j transporte l'objet o_i et elle tombe sur une case vide alors*
        - *Calculer $f(o_i)$ et $P_{drop}(o_i)$*
        - *Si $P_{drop}(o_i)$ > random-float 1.0 alors la fourmi aj dépose l'objet oi sur cette case*
      - *FinSi*
    - *FinSi*
  - *FinPour*
- *FinPour*

Fusionner les ouliers avec les clusters générés

- *Si le nombre de clusters générés = le nombre souhaité alors STOP*
- *Sinon*
  - *Introduire les agents anti-clustering pendant une durée qui vaut (T_{max} /10) itérations*
  - *Réinitialiser la variable threshold via la méthode dichotomie*
  - *Aller au « Début »*
- *FinSi*

Figure IV.4 L'algorithme de notre approche

IV.8 Conclusion

Nous avons présenté dans ce chapitre un nouvel algorithme de clustering de données basé sur les colonies de fourmis avec contrôle de l'émergence. L'approche englobe trois idées qui sont : une nouvelle stratégie adoptée par la fourmi, la modification des opérations de ramassage et de dépôt des objets et l'introduction des agents anti-clustering pour détruire l'émergence négative.

La démarche de cette approche suit trois étapes principales qui sont : Le choix approprié d'un seuil de similarité au travers plusieurs tentatives tout en utilisant la méthode dichotomie, la constitution des clusters initiaux lors de chaque tentative et finalement la fusion des outliers avec les clusters initiaux aussi lors de chaque tentative.

Dans le chapitre suivant nous présentons l'étude expérimentale qui a permis d'évaluer l'approche proposée.

CHAPITRE V: EXPÉRIMENTATION & VALIDATION

Afin d'évaluer les performances de l'approche que nous avons proposée, une étude expérimentale a été conduite en utilisant des jeux de données tests synthétiques et réels et qui représentent des distributions de données variées. Pour ce faire la première étape a été le choix des mesures d'évaluation.

V.1 Mesures d'évaluation de la qualité de clustering

Afin de pouvoir évaluer la qualité de la classification obtenue, nous avons sélectionné -parmi les mesures d'évaluation présentées dans le chapitre I deux mesures de différents types. La première évaluation est faite avec la mesure externe F-mesure. Cette mesure compare les classes résultantes de l'algorithme proposé, avec les classes fournit par les experts du domaine de chaque jeux de données. F-mesure prend des valeurs dans l'intervalle [0,1] et devrait être maximale (si F-measure = 1 ça veut dire que les classes résultantes de l'algorithme proposé sont identiques aux classes réelles du jeux de données).

Dans le cas où les classes des jeux de données ne sont pas disponibles, nous pouvons recourir à une évaluation interne. Dans notre cas, nous avons proposé d'utiliser l'indice de Davies-Bouldin pour couvrir les cas où notre algorithme s'applique sur des jeux de données sans classes connues. Cette mesure est basée sur la minimisation du rapport des dispersions intra-classe et de la séparation interclasse.

V.2 Jeux de données utilisés (Benchmarks)

Un benchmark est un banc d'essai permettant de mesurer les performances d'un système pour le comparer à d'autres. Ces bases de données sont supervisées (on connaît la classe de chaque donnée) afin de pouvoir évaluer la qualité de la classification obtenue.

Afin d'analyser cet algorithme, nous l'avons évalué sur des données réelles et sur des données synthétiques. Les jeux de données synthétiques dénotés **xdyc**, où **x** indique la dimension de l'espace des données des données et **y** signifie le nombre de clusters. Ils sont obtenus avec un générateur gaussien de clusters développé par Handl et Knowles [Handl et al, 05a]. Ce générateur génère des clusters de forme sphérique et allongée.

Les jeux de données réelles que nous avons utilisés sont issus de l'UCI Machine Learning Repository [Blake et al, 98]. Les jeux de données synthétiques et réelles sont récapitulés dans le

tableau VI.1, où **m** est le nombre total de points de données dans le jeu de données, $\mathbf{n_i}$ est le nombre de points de données appartenant au cluster **i**, **dim** est la dimension de l'espace des données et **k** le nombre de clusters (voir l'annexe A).

	Nom	**k**	**dim**	$\mathbf{n_i}$	**m**
jeux de données synthétiques	2d4c	4	2	369,471,53,230	1123
	2d10c	10	2	67,15,19,53,83,64,65,68,68,18	520
	10d4c	4	10	252,244,166,71	733
	10d10c	10	10	18,83,57,26,67,50,12,72,39,12	436
	10d20c	20	10	20,30,12,9,24,31,10,23,15,41,32,9,13,26,10,36,13,40,10,29	433
jeux de données réels	iris	3	4	50,50,50	150
	ruspini	4	2	20,23,17,15	75
	wisconsin	2	9	444,239	683
	thyroid	3	5	150,35,30	215

Tableau V.1 Jeux de données utilisées dans l'évaluation

V.3 Résultats et comparaison

Les résultats sont évalués en utilisant la F-mesure et l'indice de Davies-Bouldin (les deux indices sont détaillés dans le chapitre I). Ces résultats sont obtenus en utilisant la plateforme multi agents NETLOGO (voir *annexe-B* pour plus de détails sur NETLOGO). La comparaison est faite avec le célèbre algorithme Kmeans et l'ACO (de Deneubourg). Ce dernier est un algorithme biomimétique s'inspirant aussi du comportement des fourmis réelles. Ces algorithmes étant stochastiques, nous avons effectué 15 exécutions pour chaque algorithme sur l'ensemble des 9 bases de données afin d'obtenir une moyenne de résultats.

Les deux algorithmes sont décrits en détail dans le chapitre I et dans le chapitre II respectivement. Les opérations de ramassage et de dépôt des objets utilisées par Deneubourg sont représentées par les probabilités P_p et P_d : $P_p = (k1/k1+f)^2$ $P_d = (f/k2+f)^2$

Où f est une estimation du nombre d'objets placés dans le voisinage de la fourmi (la proportion d'éléments perçus dans le voisinage de la fourmi). k1 et k2 sont des constantes positives respectivement initialisées à 0.1 et 0.3 [Deneubourg, 90].

Le tableau V.2 présente les résultats de F-mesure obtenus par les trois algorithmes. La meilleure partition retenue est alors celle qui permet d'obtenir un résultat de F-mesure maximal.

Jeux de données	Notre algorithme	k-means	ACO
ruspini	**0,9887**	0,9047	0,9880
2d4c	0,9467	0,9700	**0,9900**
10d4c	**0,9413**	0,9140	0,7667
wisconsin	0,9353	**0,9600**	0,7033
iris	**0,8867**	0,8320	0,8800
2d10c	0,8347	0,8647	**0,9300**
10d10c	0,8180	0,8887	**0,8980**
10d20c	0,7967	0,8393	**0,9800**
thyroid	**0,7640**	0,7533	0,7167

Tableau V.2 Résultats obtenus de F-mesure

Le tableau V.2 montre clairement que notre algorithme apporte les meilleurs résultats sur la moitié des bases testées. En revanche, l'ACO donne des valeurs de f-measure plus élevées sur l'autre moitié. K-means arrive en dernière position.

La figure V.1 donne une interprétation graphique de ce tableau.

Cet ordre de performance revient à conclure que les algorithmes biomimétiques s'inspirant du comportement des fourmis sont mieux que k-means en termes de f-measue.

Notre algorithme et l'ACO ont eu des résultats très proches, mais l'ACO reste moins performant sur les bases contenant des groupes difficilement discernables, et son temps d'exécution est beaucoup plus élevé que celui obtenu par notre algorithme.

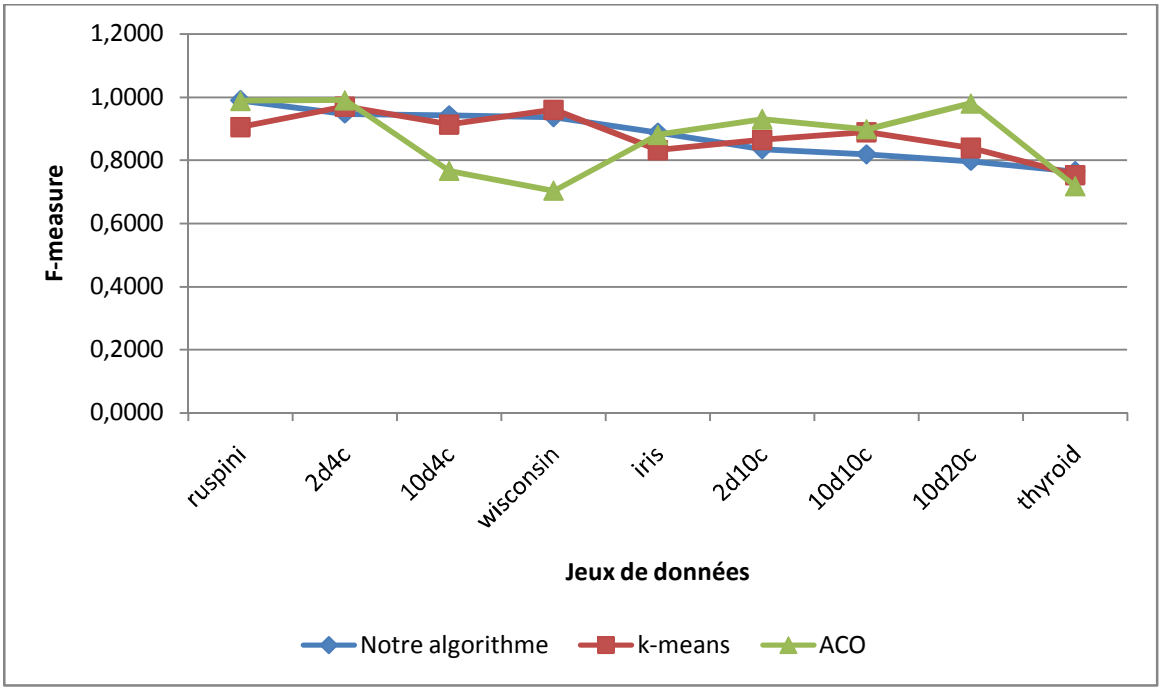


Figure V.1 Représentation graphique de résultats obtenus de F-mesure

Le tableau V.3 présente les résultats de l'indice de Davies-Bouldin obtenus par les trois algorithmes. On constate ainsi que le ratio sera d'autant plus faible que les classes seront compactes et éloignées les unes des autres. Par conséquence, le clustering de meilleure qualité sera celui qui minimisera l'indice de Davies-Bouldin (coloré en bleu).

Jeux de données	Notre algorithme	k-means	ACO
ruspini	**0,3620**	0,5040	0,3653
2d4c	0,5687	**0,4000**	0,4127
10d4c	**1,3180**	1,4120	2,6560
wisconsin	0,7847	**0,7500**	1,4960
iris	**0,7027**	0,7347	0,7280
2d10c	0,8013	0,7020	**0,5780**
10d10c	1,5073	1,2400	**0,7507**
10d20c	1,4553	1,2633	**0,7400**
thyroid	1,1993	**0,9160**	1,0467

Tableau V.3 Résultats obtenus de l'indice de Davies-Bouldin

Le tableau V.3 montre clairement que chaque algorithme apporte les meilleurs résultats sur un tiers de bases testées. Cette égalité de performance revient à conclure qu'il n'y a pas de relation entre F-measure et l'indice de Davies-Bouldin, k-means montre bien cette conclusion. Par exemple, dans le benchmark thyroid on voit que notre algorithme est meilleur que k-means en termes de F-measure (0.7640 vs 0,7533), mais k-means est meilleur que notre algorithme en termes de l'indice de Davies-Bouldin (1,1993 vs 0.1960). Même chose dans le benchmark 2d4c, mais entre k-means et l'ACO.

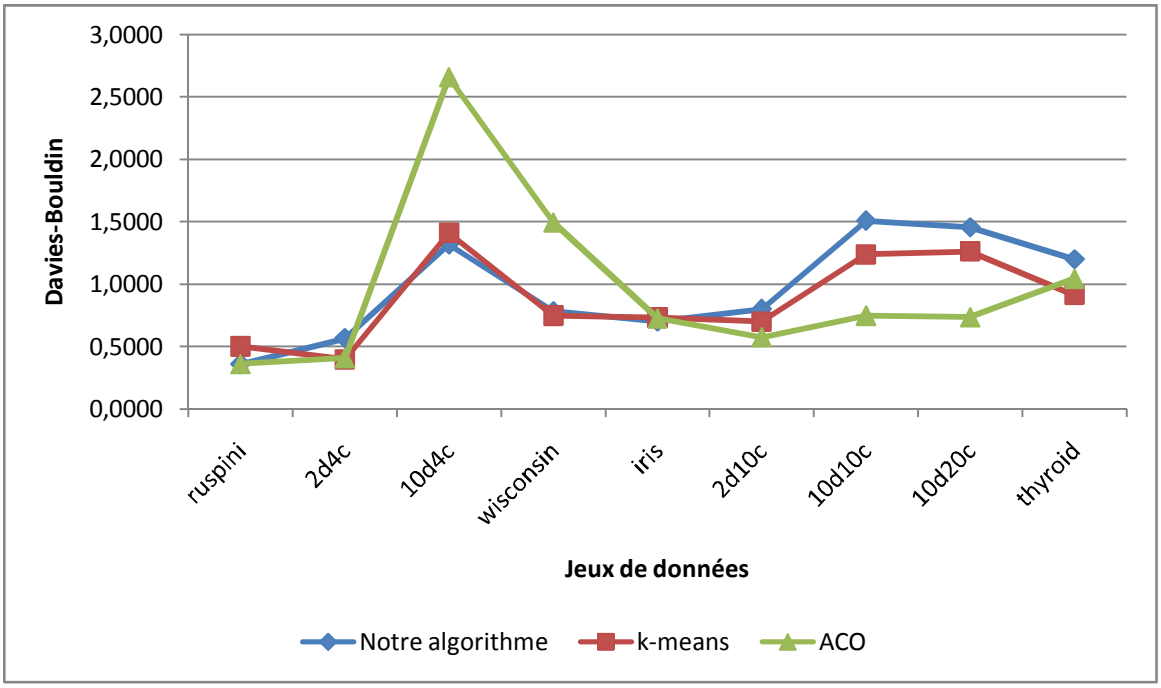


Figure V.2 Représentation graphique de résultats obtenus de l'indice de Davies-Bouldin

V.4 Comparaison entre la version classique de pick-up & drop et celle proposée

Puisque nous avons procédé au changement de pick-up & drop, il serait intéressant de montrer également les résultats avec la version classique **($P_{pick} = (k1/k1+f)^2$ $P_{drop} = (f/k2+f)^2$ où k1 = 0.1 et k2 = 0.3)**, et la version que nous proposons **($P_{pick} = (k * 100 / (k * 100 + f))^2$ $P_{drop} = (k / (k + (1 - f)))^2$ où k = 0.01)**.

Le tableau V.4 montre la moyenne de résultats de 15 exécutions, exécutées par notre algorithme.

Jeux de données	F-measure		Davies-Bouldin	
	Classique	Proposée	Classique	Proposée
ruspini	0,9866	**0,9887**	0,4237	**0,3620**
2d4c	0, 9048	**0,9467**	0,7427	**0,5687**
10d4c	0,9005	**0,9413**	1,4808	**1,3180**
wisconsin	0,9064	**0,9353**	0,8109	**0,7847**
iris	0,8008	**0,8867**	1,0066	**0,7027**
2d10c	0,7831	**0,8347**	0,8891	**0,8013**
10d10c	0,7285	**0,8180**	1,7025	**1,5073**
10d20c	0,5908	**0,7967**	1,7281	**1,4553**
thyroid	0,7285	**0,7640**	1,2004	**1,1993**

Tableau V.4 Comparaison entre la version classique de pick-up & drop et celle proposée

D'après le tableau V.4, il s'est avéré que la version proposée de pick-up & drop donnent des résultats meilleurs que ceux données par la version classique en termes de f-measure et en termes de l'indice de Davies-Bouldin (meilleur résultat coloré en bleu).

V.5 L'apport de notre approche dans le processus de data-mining

Notre approche s'inscrit dans le cadre d'extraction de connaissances à partir de données (data-mining), afin de montrer l'apport de notre approche dans le processus de data mining nous allons nous focaliser sur un jeu de données réel (Thyroid), car les clusters de données réelles sont jugés corrects par l'avis de l'expert du domaine, c'est-à-dire derrière ces données il y a une connaissance à extraire et une décision à prendre à la base de cette connaissance. L'utilisation des jeux de données synthétiques n'était que pour démonter des propriétés de notre algorithme. L'application de notre algorithme de clustering sur le jeu de données Thyroid qui contient des informations sur 215 glandes thyroïdes. Le regroupement de ces glandes essaye de **prévoir** si la glande thyroïde d'un patient appartient à la classe des glandes ayant un fonctionnement normal, à la classe des glandes ayant un hypo fonctionnement ou à la classe des glandes ayant un hyper fonctionnement.

V.6 Conclusion

Nous avons montré dans ce chapitre que les fourmis - avec contrôle de l'émergence - sont une source d'inspiration pertinente pour résoudre le problème de clustering de données à des fins de data mining. Nous avons comparé notre algorithme à deux algorithmes classiques de classification non supervisée qui sont le K-Means et l'ACO (de Deneubourg). Ce dernier est un algorithme biomimétique s'inspirant aussi du comportement des fourmis réelles.

On constate que notre algorithme donne une valeur de f-measure plus élevée que K-Means sur la majorité des bases testées, mais l'ACO donne une valeur de f-measure plus élevée sur la moitié des bases testées, et notre algorithme sur l'autre moitié, mais l'ACO reste moins performant sur les bases contenant des groupes difficilement discernables, et son temps d'exécution est beaucoup plus élevé que celui obtenu par notre algorithme.

Concernant l'indice de Davies-Bouldin, il s'est avéré que chaque algorithme apporte les meilleurs résultats sur un tiers de bases testées.

D'après les études expérimentales, il s'est avéré que notre algorithme semble apporter des résultats compétitifs par rapport à ceux auxquels on a comparés.

CONCLUSION GÉNÉRALE ET PERSPECTIVES

1 Bilan du travail

Les fourmis possèdent des comportements auto-organisés, à travers lesquels des interactions simples au niveau local permettent l'émergence d'un comportement global complexe. Cette émergence peut se produire sans qu'elle ne soit voulue, et c'est la raison pour laquelle on a adopté la technique du contrôle d'émergence.

Nous avons proposé de bénéficier du mécanisme de coopération inconsciente chez les fourmis pour mettre au point une nouvelle approche de clustering de données à des fins de data mining basé sur les colonies de fourmis avec contrôle de l'émergence.

L'approche englobe trois idées qui sont :

- Une nouvelle stratégie adoptée par la fourmi, où chaque fourmi utilise une courte mémoire locale dans laquelle elle mémorise chaque objet rencontré dans son parcours, la taille de cette mémoire est fixée à huit (une liste gérée selon la politique First In First Out). Contrairement aux méthodes classiques, où la fourmi perçoit son voisinage.
- La modification précédente exige une modification au niveau des opérations de ramassage et de dépôt des objets (pick-up & drop).
- L'introduction des agents anti-clustering, pour détruire l'émergence négative. Ces agents agissent de façon inverse par rapport au comportement de fourmis artificielles.

La démarche de cette approche suit trois étapes principales qui sont :

- Le choix approprié d'un seuil de similarité au travers plusieurs tentatives tout en utilisant la méthode dichotomie.
- La constitution des clusters initiaux lors de chaque tentative.
- La fusion des outliers avec les clusters initiaux lors de chaque tentative.

Cette approche a été testée sur des bases de données réelles (benchmarks) et les résultats obtenus sont significatifs. Nous avons comparé notre algorithme à deux algorithmes classiques de classification non supervisée qui sont le K-Means et l'ACO (de Deneubourg). Ce dernier est un algorithme biomimétique s'inspirant aussi du comportement des fourmis réelles.

D'après les études expérimentales, il s'est avéré que notre algorithme semble apporter des résultats compétitifs par rapport à ceux auxquels on a comparés.

2 Perspectives

Pour le clustering de données basé sur les colonies de fourmis avec contrôle de l'émergence, trois orientations essentielles peuvent être dégagées :

- La première consiste à améliorer l'algorithme de telle sorte qu'il peut détecter le nombre de clusters automatiquement, l'idée derrière cette amélioration est d'introduire les techniques qui mesurent la qualité de clustering, c'est-à-dire on calcule la qualité de clustering de temps en temps (au bout d'une durée limitée), et le contrôle de l'émergence s'effectue en fonction de la qualité de clustering, et non pas en fonction du nombre de clusters générés comme le cas de notre algorithme.
- La deuxième concerne à hybrider cet algorithme (tel que décrit dans le point précédent) avec k-means. L'astuce est de découper le processus de clustering en deux phases, cet algorithme joue le premier rôle qui consiste à détecter les noyaux de départ (leur nombre et leur sélection) et k-means joue le deuxième rôle. Cette hybridation principalement vise à accélérer le processus de clustering. Bien que K-Means soit un algorithme très rapide en termes de temps d'exécution, il a deux points faibles, le premier qu'il exige le nombre de clusters à priori, le deuxième se résume à la sélection des objets initiaux, cette sélection est importante, car les résultats directement dépendent de cette sélection. Finalement, on peut dire qu'avec cette hybridation:
    - Le problème lié au nombre de clusters qui doit être requis à priori est résolu.
    - Les noyaux de départ créés, vont jouer le rôle des objets initiaux du k-means.
- La troisième est liée à la possibilité de prévoir l'émergence négative. Cette prévision s'appuie sur l'injection des fourmis préventives…

Références

- [Abraham, 03] A. Abraham et V. Ramos, “Web Usage Mining Using Artificial Ant Colony Clustering and Genetic Programming”. Proc. of the Congress on Evolutionary Computation (CEC 2003), Canberra, pp. 1384- 1391, IEEE Press. 2003.
- [Ankerst et al, 99] M. Ankerst, M. Breunig, H.P. Kriegel, J. Sander. “OPTICS: Ordering Points to Identify the Clustering Structure”. In the Proc. ACM SIGMOD’99Int. Conf. on Management of Data, 1999.
- [Attar, 12] A. Attar, Thèse doctorat, “Estimation robuste des modèles de mélange sur des données distribuées”, UNIVERSITÉ DE NANTES, 2012.
- [Azzag, 04] H. Azzag, C. Guinot et G. Venturini. AntTree: “web document clustering using artificial ants”. Proceedings of the 16th European Conference on Artificial Intelligence (ECAI 04), p. 480-484, IOS Press, Valencia, Spain 2004.
- [Azzag, 05] H Azzag “Classification hiérarchique par des fourmis artificielles : applications à la fouille de données et de textes pour le Web” Thèse de Doctorat, université de Tours, 2005.
- [Berkhin, 02] P. Berkhin, “Survey of Clustering Data Mining Techniques”, Technical report, Accrue software, San Jose, California, 2002.
- [Berry et al, 06] M. Berry et M. Browne. “Lecture notes in data mining”. University of tennessee, 2006.
- [Bezdeck, 81] J. C. Bezdeck. “Pattern recognition with fuzzy objective function algorithms”. Plenum Press Ed., New York, 1981.
- [Bezdek and Pal, 98] J. C. Bezdek and N. R. Pal “Some new indexes of cluster validity”, IEEE Transactions on Systems, Man and Cybernetics, 28(3), pp. 301-315, 1998.
- [Bilmes et al, 97] J.Bilmes, A.Vahdat, W.Hsu et E.J.Im. “Empirical observations of probabilistic heuristics for the clustering problem”. Technical Report TR-97-018, International Computer Science Institute, University of California, Berkeley, CA, 1997.
- [Binitha et al, 12] S. Binitha, S. Sathya. “A Survey of Bio inspired Optimization Algorithms”. International Journal of Soft Computing and Engineering. 2012
- [Blake et al, 98] C. L. Blake and C. J. Merz. UCI repository of machine learning databases, 1998. http://www.ics.uci.edu/~mlearn/Machine-Learning.html
- [Bonabeau et al, 99] E. Bonabeau, M. Dorigo, G. Theraulaz, “Swarm Intelligence: From Natural to Artificial Systems”. New York: Oxford University Press. 1999.
- [Bonabeau, 97] E. Bonabeau et G. Theraulaz, “Auto-organisation et comportements collectifs : la modélisation des sociétés d’insectes, Auto-organisation et comportement”, Editions Hermès, 1997.
- [Cakar et al, 07] E. Cakar, M. Mnif, C. Muller-Schloer, U. Richter, and H. Schmeck “Towards a quantitative notion of self-organisation”. In Proceedings of the IEEE Congress on Evolutionary Computation (CEC2007), 4222–4229. IEEE Press. 2007.

- [Camazine, 01] S. Camazine, “Self-Organization in biological systems”. Princeton Studies in Complexity. Princeton University Press. 2001.
- [Carey et al, 07] M.J. Carey et S. Ceri. “Web DataMining”. University of Illinois at Chicago, 2007.
- [Caro et Dorigo, 98] G. Di Caro et M. Dorigo. Antnet: “Distributed stigmergetic control for communications networks”. *Journal of Artificial Intelligence Research*, 9:317-365, 1998.
- [Castro et al, 00] L. N. de Castro, F.J. Von Zuben. “An Evolutionary Immune Network for Data Clustering”. In Proc. of the IEEE SBRN (Brazilian Symposium on Artificial Neural Networks), pp. 84-89, 2000.
- [Chen et al, 04] L. Chen, X-H. Xu, and Y-X. Chen. “An adaptative ant colony clusternig algorithm”. In The Third International Conference On Machine Learning And Cybernetics, 2004.
- [Clerc, 03] M. Clerc, “Un exemple d'optimisation par essaim particulaire sans paramètre de contrôle”, Conférence OEP'03, Paris, France, 2003.
- [Cleuziou, 04] G. Cleuziou “Une méthode de classification non-supervisée pour l'apprentissage de règles et la recherche d'information”. Thèse de Doctorat, université d'Orléans, 2004.
- [Couzin et al, 05] I. Couzin, J. Krause, NR. Franks, and SA. Levin “Effective leadership and decision-making in animal groups on the move”. In Nature, 433: 513–516. 2005.
- [Davies et al, 79] D. Davies, L. Bouldin “A Cluster Separation Measure”. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1(2), 224–227, 1979.
- [De Wolf et al, 04] T. De Wolf, T. Holvolet. “Emergence versous self organisation: differents concepts but promising when combined”. Lecture Notes in Computer Science, 3464, pp.1-15. 2004.
- [Deneubourg, 90] 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, pp 356–365, 1990.
- [Deneubourg, 91] J.L Deneubourg, S. Goss, N. Franks, A. 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 (eds), SAB 90 - 1st Conference On Simulation of Adaptive Behavior: From Animals to Animats, MIT Press. 1991.
- [Devèze and Fouquin, 05] B. Devèze & M. Fouquin, “DATAMINING C4.5 – DBSCAN”. 2005.
- [Dorigo, 96] M. Dorigo, V. Maniezzo, A. Colorni, “The Ant System: Optimization by a Colony of Cooperating Agents”, IEEE Transactions on Systems, Man and Cybernetics-Part B, 1(26): p. 29-41. 1996.
- [Dréo et al, 03] J. Dréo, A. Pétrowski, P. Siarry, and E. Taillard “Métaheuristiques pour l'optimisation difficile”. Eyrolles, (2003).
- [Dréo, 04] J. Dréo, Thèse de doctorat, “Adaptation de la méthode des colonies de fourmis pour l'optimisation en variables continues”. 2004.

- [Dunn, 73] J. Dunn “A fuzzy relative of the isodata process and its use in detecting compact, wellseperated clusters”, Journal of Cybernetics, 3(3), pp. 32-57, 1973.
- [Elghazel, 07] H. Elghazel “Classification et Prévision des Données Hétérogènes : Application aux Trajectoires et Séjours Hospitaliers”. Thèse de Doctorat, Université Claude Bernard Lyon 1, 2007.
- [Emi, 06] P. Emilie, “Organisation du system immunitaire felin”, Thèse de PhD, Ecole Nationale, Lyon, France (2006).
- [Ester et al, 96] M.Ester, H.P.Kriegel, J.Sander, et X. Xu. “A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise”. In Proceeding of 2nd Int. Conference on Knowledge Discovery and Data Mining, pp. 226–231, 1996.
- [Forbes, 04] N. Forbes, “Imitation of Life: How Biology Is Inspiring Computing”. MIT Press, Cambridge, MA. 2004.
- [Forestier, 10] G. Forestier, Thèse doctorat, “Connaissances et clustering collaboratif d’objets complexes multisources”. l’Université de Strasbourg, 2010.
- [Fowlkes et Mallows, 83] E. B. Fowlkes et C. L. Mallows. “A method for comparing two hierarchical clusterings”. J. American Statistical Associationn, 78: 553–569, 1983.
- [Ganaoui and Perrot, 04] O. EL Ganaoui, M. Perrot, “Segmentation par régions: une méthode qui utilise la classification par nuées dynamiques et le principe d'hystéresis”. 2004.
- [Gardner, 70] M. Gardner. “The fantastic combinations of john conway's new solitaire game life”. Scientific American, 223:120-123, 1970.
- [Georgé, 04] J-P. Georgé, "Résolution de problèmes par émergence : étude d'un environnement de programmation émergente" PhD thesis, Université Toulouse III-Paul Sabatier, 2004.
- [Gleize, 04] M-P Gleizes “Vers la résolution de problèmes par émergence”, Habilitation à diriger des recherches de l'Université Paul Sabatier, 2004.
- [Goss, 89] S. Goss, S. Aron, J.-L.Deneubourg, J.M. Pasteels, “Self organized shortcuts in the argentine ant”, Natur wissenschaften, Vol. 76, 1989, p. 579-581.
- [Greene, 03] W.A. Greene. “Unsupervised hierarchical clustering via a genetic algorithm”. In IEEE Press, editor, Proceedings of the 2003 Congress on Evolutionary Computation, pages 998-1005, 2003.
- [Halkidi et al, 01a] M. Halkidi, Y. Batistakis, M. Vazirgiannis, “On Clustering Validation Techniques”, Intelligent Information Systems Journal, Kluwer Pulishers, vol.17, n°2-3, pp. 107-145, 2001.
- [Han et al, 06] J. Han et M. Kamber. “Data Mining: Concepts and Techniques Second Edition”. University of Illinois at Urbana-Champaign, 2006.
- [Handl et al, 05a] J. Handl and J. Knowles. “Improvements to the scalability of multiobjective clustering”. In the Proceedings of the Congress on Evolutionary Computation, Vol 3, pp. 2372-2379, 2005.

- [Handl, 03] J. Handl, “Ant-based methods for tasks of clustering and topographic mapping: extensions, analysis and comparison with alternative techniques”. Master’s Thesis, universität Erlangen-Nürnberg, Erlangen, Germany. 2003.
- [Hanoune et al, 98] M. Hanoune et F. Benabbou. “Modélisation informatique de clients douteux, en utilisant les techniques de datamining”. In Proc. IEEE Int. Conf. on Data Engineering, IEEE Computer Society Press, 1998.
- [Hatamlou et al, 11] A. Hatamlou, S. Abdullah and Z. Othman, “Gravitational Search Algorithm with Heuristic Search for Clustering Problems”, 3rd IEEE Conference on Data Mining and Optimization (DMO), 190-193, Malaysia, 2011.
- [Jain et al, 88] A. K. Jain et R.C. Dubes. “Algorithms for Clustering Data”, Prentice Hall advanced reference series, 1988.
- [Janson et al, 05] S. Janson, M. Middendorf, and M. Beekman “Honeybee Swarms: How do Scouts Guide a Swarm of Uninformed Bees? In Animal Behaviour”, 70(2): 349–358. 2005.
- [Kennedy et al, 95] J. Kennedy et R.C. Eberhart. “Particle swarm optimization”. In the Proceedings of IEEE International Conference on Neural Networks, pp 1942–1948, 1995.
- [Kumar, 00] V. Kumar, “An Introduction to Cluster Analysis for Data Mining”. Technical report, C.S. Dept. Univ. Minnesota, 2000.
- [Labroche et al, 02] N. Labroche, F.J. Richard, N. Monmarché, A. Lenoir, G. Venturini. “Modelling of the chemical recognition system of ants”. In the International Workshop on Self-Organization and Evolution of Social Behaviour, pp 283-292, 2002.
- [Langton, 90] C. Langton, “Computation at the edge of chaos, phase transition and emergent computation”. In the international conference on artificial intelligence pages 12-37, 1990.
- [Larose, 05] D. Larose, “DISCOVERING KNOWLEDGE IN DATA An Introduction to Data Mining”, 2005.
- [Law et al, 04] M. H. C. Law, A. P. Topchy and A. K. Jain, “Multiobjective Data Clustering”, In the proceeding of IEEE Computer Society Conference on Computer Vision and Pattern Recognition,Vol 2, pp 424-430, 2004.
- [Lumer et al, 94] E.D. Lumer et B. Faieta. “Diversity and adaptation in populations of clustering ants”. In the Proceedings of the Third International Conference on Simulation of Adaptive Behaviour, pages 501–508, 1994.
- [M.Ester et al, 98] M. Ester, H. Kriegel, J. Sander, M. Wimmer et X. Xu. “Incremental Clustering for Mining in a Data Warehousing Environment”. In the Proceedings of the 24th VLDB Conference New York, USA, 1998.
- [Mayers, 10] A. Mayers, “Segmentation basée sur la densité DBSCAN”. 2010.

- [Merwe et al, 03] D. Merwe et A.Engelbrecht. “Data clustering using particle swarm optimization”. In the Proceedings of IEEE Congress on Evolutionary Computation, Canbella, Australia. pp. 215-220, 2003.
- [Mogul, 06] JC Mogul. “Emergent (mis)behavior vs. complex software systems”. In SIGOPS Operating Systems Review, 40(4): 293–304. 2006.
- [Monmarché, 99] N. Monmarché. “On data clustering with artificial ants”. In the Workshop on Data Mining with Evolutionary Algorithms, pp 23-26, 1999.
- [Nasaroui et al, 02] 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, pp 711–716, 2002.
- [Omran, 05] M.G.H.Omran. “Partical Swarm optimisation methods for pattern recognition and image processing”. Thèse de Doctorat, Université de Pretoria, 2005.
- [Ou et Lin, 06] C.Ou et W.Lin, “Comparison between PSO and GA for Parameters Optimization of PID controller”. China, Proceedings of the IEEE. 2006.
- [Parunak, 97] H. Parunak, “Go to the Ant: Engineering Principles from Natural Multi-Agent Systems”. Annals of Operations Research 75 69-101 Special Issue on Artificial Intelligence and Management Science, 1997.
- [Pedrycz, 05] W. Pedrycz. “Clustering and Fuzzy Clustering”. chapitre1, In Knowledge Based Clustering, Wiley & Sons, Hoboken, 2005.
- [Picarougne et al, 05] F. Picarougne, H. Azzag, Gilles Venturini, et Christiane Guinot. “A new approach of data clustering using a flock of agents”. Evolutionary Computation, 2005. à paraître.
- [Proctor et Winter, 98] G. Proctor et C. Winter. “Information flocking: Data visualisation in virtual worlds using emergent behaviours”. In J.-C. Heudin, editor, Proc. 1st Int. Conf. Virtual Worlds, VW, volume 1434, pages 168–176. Springer-Verlag, 1998.
- [Ramos, 02] V. Ramos, F. Muge, F. and P. Pina, “Self-organized data and image retrieval as a consequence of inter-dynamic synergistic relationships in artificial ant colonies”, Vol. 87, IOS Press. 2002.
- [Rendon et al, 11] E. Rendon, I. Abundez, A. Arizmendi, “Internal versus External cluster validation indexes”. International Journal of Computers and Communications 5(1), 27–34 (2011).
- [Rogovschi, 09] N. Rogovschi “Classication à base de modèles de mélanges topologiques des données catégorielles et continues”. Thèse de Doctorat, Université Paris 13 - Insitut Galilée, 2009.
- [Roitt, 90] I. Roitt, “Immunologie”. Editions Pradel, 1990.
- [Rosenberg, 07] A. Rosenberg and J. Hirschberg. V-measure “A conditional entropy-based external cluster evaluation measure”. In EMNLP-CoNLL’07, pages 410–420, 2007.

- [Rousseeuw, 87] P .Rousseeuw, J. Silhouettes: “a Graphical Aid to the Interpretation and Validation of Cluster Analysis, Journal of Computational and Applied Mathematics”, 20, pp. 53-65, 1987.
- [Sander et al, 98] J.Sander, M.Ester, H.P. Kriegel, X. Xu. “Density-Based Clustering in Spatial Data sets: The Algorithm GDBSCAN and Its Applications”. Data Mining and Knowledge Discovery 2, Kluwer Academic Publishers, 169–194, 1998.
- [Scheidler, 10] A. Scheidler, “Self-Organized Specialization and Controlled Emergence in Organic Computing Systems”. Thèse de Doctorat, Université de Leipzig, 2010.
- [Steinbach et al, 03] M. Steinbach, L. Ertöz, et V. Kumar. “Challenges of Clustering High Dimensional Data”. In L. T. Wille, editor, New Vistas in Statistical Physics– Applications in Econophysics, Bioinformatics, and Pattern Recognition. Springer-Verlag, 2003.
- [Sugiyama, 07] M. Sugiyama “Dimensionality Reduction of Multimodal Labeled Data by Local Fisher Discriminant Analysis”, Journal of Machine Learning Research 8 (2007) 1027-1061.
- [Sumpter et al, 08] D. Sumpter, J. Krause, R. James, I .Couzin, and A. Ward “Consensus Decision Making by Fish”. In Current Biology, 18(22): 1773–1777. 2008.
- [Xiao et al, 03] X.Xiao, E.R.Dow, R.C.Eberhart, Z.Ben Miled, et R.J.Oppelt. “Gene clustering using self-organizing maps and particle swarm optimization”. In the Proceedings of Second IEEE International Workshop on High Performance Computational Biology, Nice, France. 2003
- [Xu et al, 98] X. Xu, M. Ester, H.P. Kriegel et J. Sander. “A Nonparametric Clustering Algorithm for Knowledge Discovery in Large Spatial Datasets”. In Proc. IEEE Int. Conf. on Data Engineering, IEEE Computer Society Press, 1998.

ANNEXES

Annexe A : Descriptif des jeux de données réelles utilisés

Cette annexe présente les principales bases de tests sur lesquelles ont été testés les algorithmes présentés dans le chapitre expérimentation & validation. Elles ont été choisies pour démontrer l'adaptabilité de l'algorithme aux types de données et à la taille des problèmes.

A.1 Iris

"Les iris de Fisher" sont des données proposées en 1933 par le statisticien Ronald Aylmer Fisher comme données de référence pour l'analyse discriminante et la classification. Les données correspondent à 3 espèces de fleurs (Iris setosa, Iris virginica, Iris versicolor). Ce jeu de données contient 150 instances réparties à trois classes (50 par classe). Les variables mesurées sont la longueur et la largeur des sépales, la longueur et la largeur des pétales (4 dimensions). Toutes ces variables sont exprimées en centimètres.

A.2 Ruspini

Les données Ruspini consistent en 75 points (2 dimensions) répartis en quatre groupes. Ce jeu de données est populaire pour illustrer les techniques de clustering.

A.3 Wisconsin

Ce jeu de données provient initialement de l'hopital de l'Université du Wisconsin à Madison et ont été collectées par Wolberg et Mangasarian. Ces données contiennent les informations médicales de 683 cas cliniques relatifs au cancer du sein classés comme bénin ou malin (2 classes). Les cas cliniques sont décrits par 9 attributs numériques (dimensions).

1. Clump Thickness
2. Uniformity of Cell Size
3. Uniformity of Cell Shape
4. Shape Marginal Adhesion
5. Single Epithelial Cell Size
6. Bare Nuclei
7. Bland Chromatin
8. Normal Nucleoli
9. Mitoses

A.4 Thyroid

Le jeu de données Thyroid contient des informations sur 215 glandes thyroïdes. Le regroupement de ces glandes essaye de prévoir si la thyroïde d'un patient appartient à la classe euthyroidism, à l'hypothyroïdisme ou à l'hyperthyroïdisme. Chaque glande est décrite par 5 attributs numériques :

1. T3-resin uptake test. (%)
2. Total Serum thyroxin as measured by the isotopic displacement method.
3. Total serum triiodothyronine as measured by radioimmuno assay.
4. Basal thyroid-stimulating hormone (TSH) as measured by radioimmuno assay.
5. Maximal absolute difference of TSH value after injection of 200 micro grams of thyrotropin-releasing hormone as compared to the basal value.

[Blake et al, 98] [Sugiyama, 07]

Annexe B : NetLogo (plateforme de validation)

B.1 Qu'est-ce que NetLogo?

NetLogo est un environnement de modélisation programmable permettant de simuler des phénomènes naturels et sociaux. Il a été créé par Uri Wilenski en 1999 et son développement est poursuivi de manière continue par le Center for Connected Learning and Computer-Based Modeling.

NetLogo convient tout particulièrement à la modélisation de systèmes complexes évoluant au cours du temps. Les « modélisateurs » peuvent donner des instructions à des centaines ou des milliers d'« agents » opérant indépendamment les uns des autres. Ce qui permet d'explorer les liens entre les comportements des individus à leur niveau et les schémas généraux (comportements de groupe ou de masse) qui émergent des$ interactions entre de nombreux individus.

NetLogo permet aux étudiants d'ouvrir des modèles et de « jouer » avec leurs simulations pour explorer leurs comportements en faisant varier diverses conditions (paramètres). C'est aussi un environnement de programmation permettant aux étudiants, aux enseignants et aux développeurs de cours de formation de créer leurs propres modèles. NetLogo est suffisamment simple pour permettre aux étudiants et aux enseignants d'utiliser les simulations sans trop de difficultés, voire d'en concevoir

eux-mêmes. Mais il est aussi assez performant pour servir d'outil puissant pour des chercheurs dans de nombreux domaines.

NetLogo est accompagné d'une documentation complète et de nombreux tutoriaux. Il est également livré avec une Bibliothèque de modèles qui comprend une grande collection de simulations fonctionnelles pouvant être utilisées telles quelles ou modifiées selon les fantaisies de l'utilisateur. Ces simulations couvrent de nombreux domaines des sciences naturelles et sociales, y compris la biologie et la médecine, la physique et la chimie, les mathématiques et l'ingénierie informatique ainsi que l'économie et la psychologie sociale. De nombreuses sessions d'apprentissage à base de modèles utilisant NetLogo sont actuellement en cours de développement.

NetLogo offre aussi un outil puissant de simulations participatives au niveau de la classe appelé HubNet. Grâce à l'utilisation d'un réseau d'ordinateurs ou de terminaux «manuels» tels que les calculatrices graphiques de Texas Instruments, chaque élève d'une classe peut contrôler en réseau l'un des agents de la simulation.

NetLogo représente la nouvelle génération de toute une série de langages de modélisations multi-agents qui a débuté avec StarLogo. Il a été construit à partir des bases fournies par notre logiciel StarLogo auquel il apporte toute une série de nouvelles fonctionnalités significatives ainsi qu'un langage et une interface utilisateur entièrement remaniés. NetLogo étant écrit en Java, il peut tourner sur tous les systèmes majeurs (Mac, Windows, Linux et autres). Il fonctionne en tant qu'application indépendante. Les modèles peuvent même être sauvegardés sous forme d'applets Java et tourner dans tous les navigateurs internet modernes.

B.2 Fonctionnalités

Consultez la liste ci-dessous pour prendre connaissance des fonctionnalités offertes par NetLogo.

- Système :
  - ✓ Multi-plateforme: tourne sur Mac, Windows, Linux, et autres
- Langage :
  - ✓ Entièrement programmable
  - ✓ Structure du langage simple
  - ✓ Dialecte Logo étendu pour supporter les agents
  - ✓ Agents mobiles (les tortues) se déplaçant sur une grille formée d'agents stationnaires (les patches)

- ✓ Création de liens entre les tortues pour former des agrégats, des réseaux et des graphes
- ✓ Important vocabulaire de primitives intégrées au langage
- ✓ Calculs en virgule flottante double précision (IEEE 754)
- ✓ Fonctionnement absolument identique sur toutes les plateformes

- Environnement :
  - ✓ Observation des modèles en 2D ou en 3D
  - ✓ Formes vectorielles redimensionnables et pivotantes
  - ✓ Étiquetage des tortues et des patches
  - ✓ Centre de commande pour un pilotage interactif (en direct) de la simulation
  - ✓ Interface pouvant recevoir des boutons, des curseurs, des commutateurs, des traceurs de courbes, des sélecteurs, des moniteurs, des boîtes de texte, des notes, des zones de sortie
  - ✓ Variateur de vitesse pour accélérer ou ralentir la simulation
  - ✓ Système de sortie graphique puissant et souple
  - ✓ Panneau d'informations pour annoter les modèles
  - ✓ HubNet: simulation participative utilisant des machines en réseau
  - ✓ Moniteurs d'agent pour inspecter et contrôler les agents
  - ✓ Fonctions d'exportation et d'importation (exportation des données, sauvegarde et restauration de l'état d'un modèle, création et enregistrement d'une animation)
  - ✓ Outils BehaviourSpace (espace de comportements) utilisé pour collecter des données provenant de plusieurs sessions de simulations
  - ✓ Modélisation de systèmes dynamiques
- Web:
  - ✓ Modèles enregistrables sous forme d'applets pouvant ensuite être intégrés dans des pages web (note: certaines fonctionnalités ne sont pas disponibles dans les applets, telles que certaines extensions et la visualisation 3D) [Manuel de l'utilisateur NetLogo]

Annexe C : Résultats obtenus sur 15 exécutions de chaque algorithme

Ces algorithmes étant stochastiques, nous avons effectué 15 exécutions pour chaque algorithme sur l'ensemble des 9 bases de données afin d'obtenir une moyenne de résultats (coloré en bleu). Les meilleurs résultats et les mauvais résultats de chaque tableau sont mentionnés dans la première et la dernière ligne respectivement (en gras).

Notre Algo ruspini	
Fmeasure	Davies-Bouldin
1,00	**0,35**
1,00	0,35
1,00	0,35
1,00	0,35
1,00	0,35
1,00	0,35
1,00	0,35
1,00	0,35
1,00	0,35
1,00	0,35
1,00	0,35
1,00	0,35
1,00	0,35
1,00	0,35
1,00	0,35
0,83	**0,53**
0,9887	0,3620

K-means ruspini	
Fmeasure	Davies-Bouldin
1,00	**0,35**
1,00	0,35
1,00	0,35
1,00	0,35
1,00	0,35
1,00	0,35
1,00	0,35
1,00	0,35
0,83	0,53
0,82	0,58
0,82	0,58
0,81	0,68
0,78	0,77
0,76	0,79
0,75	**0,83**
0,9047	0,5040

ACO ruspini	
Fmeasure	Davies-Bouldin
1,00	**0,35**
1,00	0,35
1,00	0,35
1,00	0,35
1,00	0,35
1,00	0,35
1,00	0,35
1,00	0,35
1,00	0,35
1,00	0,35
1,00	0,35
1,00	0,35
1,00	0,35
1,00	0,35
1,00	0,35
0,82	**0,58**
0,9880	0,3653

Notre Algo iris	
Fmeasure	Davies-Bouldin
0,92	**0,67**
0,92	0,67
0,92	0,67
0,91	0,67
0,91	0,67
0,91	0,67
0,90	0,68
0,90	0,68
0,90	0,68
0,90	0,69
0,90	0,69
0,88	0,70
0,87	0,70
0,87	0,71
0,69	**0,99**
0,8867	0,7027

K-means iris	
Fmeasure	Davies-Bouldin
0,89	**0,66**
0,89	0,66
0,89	0,66
0,89	0,66
0,89	0,66
0,89	0,66
0,89	0,66
0,89	0,66
0,89	0,66
0,89	0,66
0,89	0,66
0,69	0,88
0,67	0,95
0,67	0,95
0,66	**0,98**
0,8320	0,7347

ACO iris	
Fmeasure	Davies-Bouldin
0,88	**0,72**
0,88	0,72
0,88	0,72
0,88	0,73
0,88	0,73
0,88	0,73
0,88	0,73
0,88	0,73
0,88	0,73
0,88	0,73
0,88	0,73
0,88	0,73
0,88	0,73
0,88	0,73
0,88	0,73
0,88	**0,73**
0,8800	0,7280

Notre Algo thyroid	
Fmeasure	Davies-Bouldin
0,83	**1,15**
0,83	1,15
0,82	1,15
0,81	1,15
0,81	1,16
0,81	1,19
0,80	1,20
0,80	1,21
0,78	1,21
0,77	1,21
0,73	1,23
0,71	1,24
0,68	1,24
0,65	1,25
0,63	**1,25**
0,7640	1,1993

K-means thyroid	
Fmeasure	Davies-Bouldin
0,86	**0,87**
0,85	0,87
0,84	0,87
0,84	0,87
0,84	0,87
0,84	0,89
0,84	0,91
0,80	0,92
0,68	0,95
0,66	0,95
0,65	0,95
0,65	0,95
0,65	0,95
0,65	0,95
0,65	**0,97**
0,7533	0,9160

ACO thyroid	
Fmeasure	Davies-Bouldin
0,73	**1,00**
0,73	1,00
0,73	1,00
0,73	1,00
0,73	1,00
0,72	1,03
0,72	1,03
0,72	1,03
0,72	1,03
0,71	1,05
0,71	1,05
0,70	1,10
0,70	1,11
0,70	1,12
0,70	**1,15**
0,7167	1,0467

Notre Algo wisconsin	
Fmeasure	Davies-Bouldin
0,94	**0,78**
0,94	0,78
0,94	0,78
0,94	0,78
0,94	0,78
0,94	0,78
0,94	0,78
0,94	0,78
0,93	0,79
0,93	0,79
0,93	0,79
0,93	0,79
0,93	0,79
0,93	0,79
0,93	**0,79**
0,9353	0,7847

K-means wisconsin	
Fmeasure	Davies-Bouldin
0,96	**0,75**
0,96	0,75
0,96	0,75
0,96	0,75
0,96	0,75
0,96	0,75
0,96	0,75
0,96	0,75
0,96	0,75
0,96	0,75
0,96	0,75
0,96	0,75
0,96	0,75
0,96	0,75
0,96	**0,75**
0,9600	0,7500

ACO wisconsin	
Fmeasure	Davies-Bouldin
0,73	**1,43**
0,73	1,43
0,73	1,43
0,72	1,47
0,71	1,47
0,71	1,47
0,71	1,52
0,71	1,52
0,70	1,52
0,69	1,52
0,69	1,52
0,68	1,52
0,68	1,54
0,68	1,54
0,68	**1,54**
0,7033	1,4960

Notre Algo 2d4c	
Fmeasure	Davies-Bouldin
0,97	**0,40**
0,97	0,42
0,96	0,42
0,96	0,43
0,96	0,43
0,96	0,44
0,96	0,46
0,96	0,50
0,95	0,55
0,95	0,55
0,95	0,58
0,95	0,66
0,91	0,79
0,90	0,91
0,89	**0,99**
0,9467	0,5687

K-means 2d4c	
Fmeasure	Davies-Bouldin
0,97	**0,40**
0,97	0,40
0,97	0,40
0,97	0,40
0,97	0,40
0,97	0,40
0,97	0,40
0,97	0,40
0,97	0,40
0,97	0,40
0,97	0,40
0,97	0,40
0,97	0,40
0,97	0,40
0,97	0,40
0,97	**0,40**
0,9700	0,4000

ACO 2d4c	
Fmeasure	Davies-Bouldin
0,99	**0,41**
0,99	0,41
0,99	0,41
0,99	0,41
0,99	0,41
0,99	0,41
0,99	0,41
0,99	0,41
0,99	0,41
0,99	0,41
0,99	0,41
0,99	0,41
0,99	0,42
0,99	0,42
0,99	0,42
0,99	**0,42**
0,9900	0,4127

Notre Algo 10d4c	
Fmeasure	Davies-Bouldin
0,96	**1,21**
0,96	1,22
0,96	1,23
0,96	1,23
0,96	1,23
0,96	1,24
0,96	1,24
0,96	1,24
0,96	1,25
0,95	1,27
0,95	1,28
0,94	1,30
0,94	1,31
0,91	1,52
0,79	**2,00**
0,9413	1,3180

K-means 10d4c	
Fmeasure	Davies-Bouldin
0,98	**1,19**
0,98	1,19
0,98	1,19
0,98	1,19
0,98	1,19
0,98	1,19
0,98	1,19
0,98	1,19
0,98	1,19
0,98	1,19
0,79	1,84
0,79	1,85
0,78	1,86
0,78	1,86
0,77	**1,87**
0,9140	1,4120

ACO 10d4c	
Fmeasure	Davies-Bouldin
0,74	**2,38**
0,75	2,38
0,75	2,49
0,76	2,49
0,75	2,56
0,75	2,56
0,76	2,67
0,77	2,69
0,78	2,70
0,73	2,75
0,80	2,77
0,79	2,78
0,79	2,80
0,80	2,85
0,78	**2,97**
0,7667	2,6560

Notre Algo 2d10c	
Fmeasure	Davies-Bouldin
0,94	**0,68**
0,89	0,74
0,87	0,74
0,86	0,74
0,85	0,76
0,85	0,79
0,85	0,80
0,85	0,81
0,84	0,82
0,83	0,82
0,81	0,84
0,80	0,85
0,78	0,86
0,76	0,87
0,74	**0,90**
0,8347	0,8013

K-means 2d10c	
Fmeasure	Davies-Bouldin
0,95	**0,61**
0,93	0,63
0,93	0,63
0,92	0,66
0,92	0,66
0,91	0,67
0,88	0,69
0,88	0,70
0,87	0,70
0,87	0,70
0,86	0,71
0,83	0,77
0,76	0,77
0,73	0,79
0,73	**0,84**
0,8647	0,7020

ACO 2d10c	
Fmeasure	Davies-Bouldin
0,93	**0,57**
0,93	0,57
0,93	0,57
0,93	0,58
0,93	0,58
0,93	0,58
0,93	0,58
0,93	0,58
0,93	0,58
0,93	0,58
0,93	0,58
0,93	0,58
0,93	0,58
0,93	0,58
0,93	**0,58**
0,9300	0,5780

Notre Algo 10d10c	
Fmeasure	Davies-Bouldin
0,92	**1,25**
0,86	1,30
0,86	1,35
0,86	1,39
0,85	1,41
0,85	1,41
0,82	1,53
0,82	1,53
0,81	1,54
0,80	1,58
0,79	1,63
0,77	1,65
0,76	1,65
0,75	1,68
0,75	**1,71**
0,8180	1,5073

K-means 10d10c	
Fmeasure	Davies-Bouldin
0,96	**1,02**
0,96	1,02
0,95	1,04
0,95	1,08
0,94	1,09
0,92	1,15
0,91	1,15
0,89	1,16
0,88	1,22
0,86	1,24
0,86	1,41
0,85	1,43
0,83	1,45
0,79	1,51
0,78	**1,63**
0,8887	1,2400

ACO 10d10c	
Fmeasure	Davies-Bouldin
0,90	**0,75**
0,90	0,75
0,90	0,75
0,90	0,75
0,90	0,75
0,90	0,75
0,90	0,75
0,90	0,75
0,90	0,75
0,90	0,75
0,90	0,75
0,90	0,75
0,90	0,75
0,89	0,75
0,89	0,75
0,89	**0,76**
0,8980	0,7507

Notre Algo 10d20c	
Fmeasure	Davies-Bouldin
0,85	**1,29**
0,85	1,30
0,84	1,31
0,84	1,31
0,83	1,35
0,83	1,36
0,82	1,37
0,81	1,45
0,79	1,47
0,78	1,49
0,78	1,53
0,75	1,60
0,74	1,64
0,73	1,67
0,71	**1,69**
0,7967	1,4553

K-means 10d20c	
Fmeasure	Davies-Bouldin
0,93	**1,10**
0,90	1,15
0,88	1,15
0,88	1,15
0,88	1,17
0,87	1,22
0,85	1,22
0,85	1,25
0,83	1,25
0,82	1,25
0,80	1,32
0,79	1,36
0,78	1,39
0,78	1,43
0,75	**1,54**
0,8393	1,2633

ACO 10d20c	
Fmeasure	Davies-Bouldin
0,98	**0,74**
0,98	0,74
0,98	0,74
0,98	0,74
0,98	0,74
0,98	0,74
0,98	0,74
0,98	0,74
0,98	0,74
0,98	0,74
0,98	0,74
0,98	0,74
0,98	0,74
0,98	0,74
0,98	**0,74**
0,9800	0,7400