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

UNIVERSITE MOHAMED BOUDIAF - M'SILA
FACULTE DES MATHEMATIQUES ET DE L'INFORMATIQUE
DÉPARTEMENT D'INFORMATIQUE

MEMOIRE de fin d'étude

Présenté pour l'obtention du diplôme de MASTER

Domaine : Mathématiques et Informatique

Filière : Informatique

Spécialité : Informatique Décisionnelle et Optimisation

Par : Arbia Djamila

SUJET

Métaheuristiques appliquées à la classification non supervisée de données

Soutenu publiquement le : 13 /07/2019 devant le jury composé de :

Dr. LAKHAL Ayat	**Université de M'sila Président**
Dr. LAMICHE Chaabane	**Université de M'sila Rapporteur**
Dr.DEBBI Imad Eddine	**Université de M'sila Examinateur**

Promotion : 2018 /2019

SOMMAIRE

Chapitre 4 : Réalisation et expérimentation

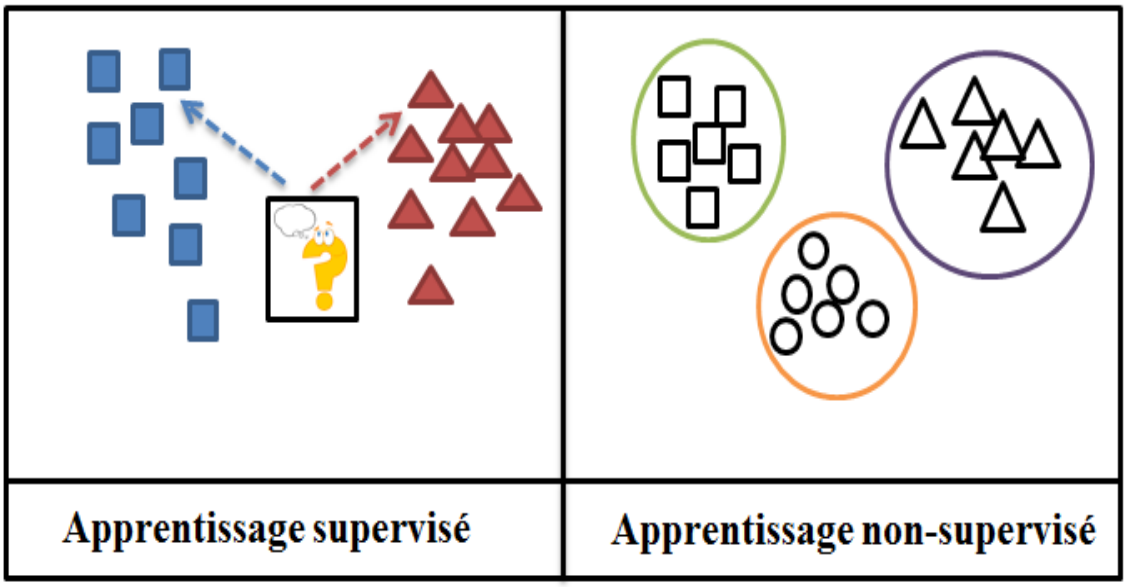
Introduction generale

Comme nous savons les données ont un intérêt considérable et avec le temps croissant dans tous les secteurs et surtout économiques et industriels, de nouvelles techniques sont apparues pour exploreret exploiter ces quantités importantes de données qui peuvent être de différentes sources et de différents types. L'une des techniques les plus répandues est le datamining qui permet d'extraire des connaissances à partir de données afin de créer un modèle intéressant et compréhensible. Le datamining se compose de taches prédictives, telles que la classification supervisée, et de taches descriptives, telles que la classification non supervisée.

La classification des données est une téchnique importante dans le domaine de l'analyse de données appliquée dans de nombreux domaines scientifiques tels que l'imagerie,la biologie, le marketing …etc, elle inclut des algorithmes et des méthodes pour regrouper ou classifier des objets, selon un critere de similarité. Les objets peuvent etre représentés soit par des vecteurs de mésures soit par des points dans un espace multidimensionnel. Dés le départ il est nécessaire de différencier la classification non supervisée et la classification supervisée ou analyse discriminante. La classification supervisée consiste á construire des règles de décision en se basant sur un ensemble de données pour lesquelles les étiquettes des classes sont connues a priori. Le but de la classification non supervisée est de trouver une organisation des données cohérente et valide, qui puisse mettre en évidence les vraies structures dans un ensemble de données sans aucune connaissance a priori sur les données traitées.

L'objectif de cette mémoire est l'optimisation de problème de la classification non supervisee des donnees par les algorithmes métaheuristiques à population telle que la méthode BBO « Biogeography-Based Optimization » et GA « l'algorithme génétique».

Ce travail comprend quatre chapitres qui nous permettent de présenter les différents aspects de notre travail.


Apprentissage supervisé
Apprentissage non-supervisé

CHAPITRE 1

Classification non supervisée de données

1.1.Introduction

De nos jours, la classification non-supervisée a trouvé différentes utilisations dans des domaines variés comme : la recherche et l'extraction d'information, dans des applications sur des données spatiales, par exemple, SIG (Système d'Information Géographique) ou sur des données astronomiques, l'analyse des données hétérogènes et séquentielles, les applications Web, l'analyse ADN en bioinformatique, etc [17]. Dans ce chapitre on va presenter le principe de base de la classfication non suppervisée et les différentes methodes classiques pour cette tache.

1.2 Cadre formel du clustering

La classification non supervisée (*clustering*) est une tâche de fouille de données qui a pour objectif de regrouper des données en un ensemble de classes ou clusters, de façon à ce que les objets d'une même classe aient une forte similarité entre eux et diffèrent fortement des objets des autres classes. La classification non supervisée est souvent vue comme un problème d'optimisation, consiste a trouver une partition des objets qui optimise un critère donné [09].

On considère un ensemble de n objets X = $\{x_1,\ldots\ldots,x_n\}$ ainsi qu'une matrice de dissimilarité D sur cet ensemble, telle que $d(x_i, x_j)$ représente la dissimilarité entre les deux objets x_i, x_j . La matrice D est de taille n × n et a valeurs dans [0 , 1]. Lorsque les données sont uniquement décrites par une telle matrice, on parle parfois de données relationnelles. Nous verrons, par la suite, que la plupart des méthodes de classification non-supervisées utilisent une description vectorielle des données. On parle alors de "données objets" et on considère un ensemble fini $V = v_1, \ldots., v_n$ de variables descriptives, telles que $v_j(x_i)$ désigne la valeur de l'objet $x_i \in X$ pour la variable $v_j \in V$.

La tâche de clustering permet de générer un ensemble de t clusters C =$c_1, \ldots.. c_t$ tel que chaque cluster Ca est un sous-ensemble de X (Ca $\subset$ X) et l'union des clusters couvre l'ensemble des objets de départ [10].

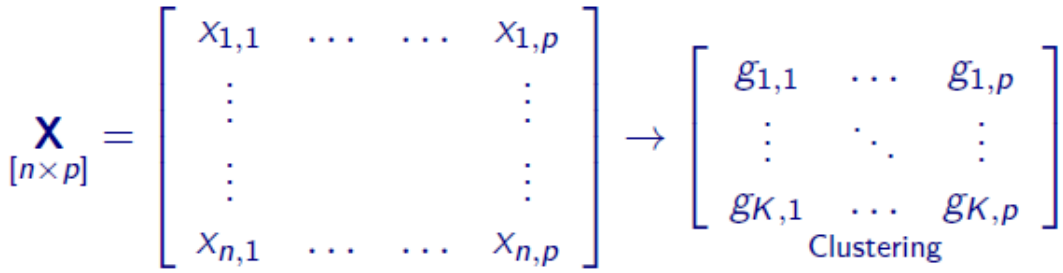


Figure 1.1 : Matrice de données et clustering.

Notons ici qu'un cluster est une agrégation des points dans l'espace d'essai tel que la distance entre deux points quelconques dans un cluster est inférieure a la distance entre n'importe quel point de ce cluster et n'importe quel point qui ne se trouve pas dans ce cluster. La mesure de similarité est une fonction mathématique qui nous permet de regrouper les objets en classes. Les metriques de distance les plus utilisées dans la littérature sont [17]:

a - La distance de Minkowski : avec p un entier positif non nul.

$$d(i,j) = \sqrt[p]{\sum_{r=1}^{M} |x_{ir} - x_{jr}|^p}$$

b- La distance euclidienne est la plus connue et la plus utilisée. Elle peut être vue comme un cas particulier de la distance de Minkowski pour p = 2 :

$$d(i,j) = \sqrt{\sum_{r=1}^{M} |x_{ir} - x_{jr}|^2}$$

1.3 Principales étapes du clustering

Le processus de clustering se divise en trois étapes essentielles :

1.3.1 Etape de préparation des données

L'étape de la préparation des données consiste à sélectionner et/ou pondérer ces variables,voire à créer de nouvelles variables, afin de mieux discriminer entre eux les objets à traiter.En effet les variables ne sont pas nécessairement toutes pertinentes : certaines peuvent être redondantes et d'autres non-pertinentes pour la tâche ciblée. Ce problème de sélection de variables a été largement étudié en classification supervisée mais reste très ouvert pour une

approche non-supervisée, dans une perspective non supervisée peu de travaux concernent la sélection de variables, principalement parce que, sans les étiquettes de classe, il est difficile d'évaluer la pertinence d'une variable. Cette difficulté peut être contournée en effectuant une première étape de clustering à partir de l'ensemble des variables, puis de considérer chaque cluster comme une classe et de réitérer ce processus. De fait, cette technique se place parmi les approches "enveloppe" puisqu'elle est fortement dépendante de l'algorithme de clustering et des paramètres utilisés (nombre de clusters, etc.). Récemment, des approches "filtres" ont été envisagées dans ce contexte non-supervisé [08].

1.3.2 Choix de l'algorithme de clustering

Dans cette étape on va choisir l'algorithme de classification la plus adapté pour le partionnement de données. Ce choix est très lié au type de données à traiter.

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.) [08].

1.3.3 Etape d'exploitation des résultats de l'algorithme (clusters) La tentation est rande, pour un non-spécialiste, de considérer comme "acquis" le résultat d'un processus de clustering. Autrement dit, les clusters obtenus ne sont généralement ni remis en cause ni évalués en terme de disposition relative, dispersion, orientation, séparation, densité ou stabilité. Pourtant, il est sans aucun doute utile de distinguer les classes pertinentes obtenues, des autres. De même, cette étape d'analyse permet d'envisager le recours à une autre approche de clustering plus adaptée. Deux situations sont possibles : soit la tâche de clustering s'inscrit dans un traitement global d'apprentissage, soit les clusters générés par clustering constituent un résultat final [08].

Dans le premier cas, l'analyse des clusters obtenus (mesures statistiques de qualité) peut aider à orienter le traitement suivant. Une description des clusters n'est pas nécessaire dans cette situation. En revanche, dans le cas où le clustering constitue à lui seul un processus global de découverte de classes, l'exploitation des clusters pour une application donnée passe par une description de ces derniers. Lorsque les objets se présentent sous la forme d'une matrice de (dis)similarité, il existe peu de méthodes pour décrire les classes (méloïdes, k objets

représentatifs et mesures de cohésion). Lorsque les objets sont décrits par un ensemble de variables, on peut avoir recours à des méthodes de description des classes [08].

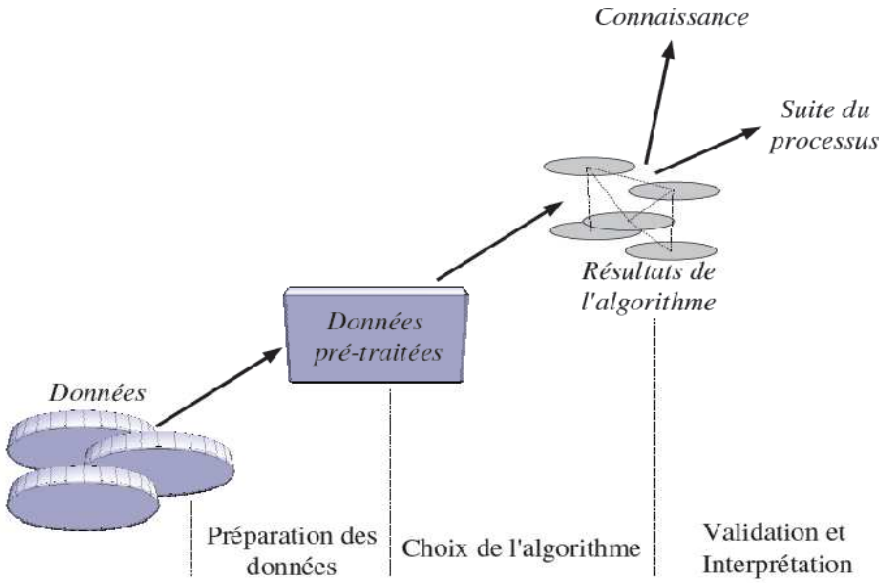


Figure1.2 : Les différentes étapes du clustering.

1.4 Strategies de clustering

Ainsi par rapport à la méthode utilisée pour définir les classes, les algorithmes peuvent être classifiés de façon générale dans les catégories suivantes [01] :

1.4.1 La classification hiérarchique

La classification hiérarchique construit une hiérarchie de clusters, ou autrement dit, un arbre de clusters, connue aussi sous le nom de dendogramme. Une telle approche permet d'explorer les données à travers différents niveaux de granularité. Les méthodes de la classification hiérarchique sont divisées en deux types d'approches : ascendante (CHA), dite agglomérative et descendante (CHD), dite divisive[17].

- La première (ascendante) : C’est la plus couramment utilisée consiste, à construire l'hiérarchie à partir des objets (au départ on a un objet par classe), puis à fusionner les classes les plus proches, afin de n'en obtenir plus qu'une seule contenant tous les objets.

- La seconde (descendante) : moins utilisée, consiste à construire l'hiérarchie à partir d'une seule classe regroupant tous les objets, puis à partager celle-ci en deux groupes. Cette opération est répétée à chaque itération jusqu'à ce que toutes les classes soient réduites à des singletons. Le processus continu jusqu'à ce qu'un critère d'arrêt (d'habitude c'est le nombre k de clusters demandé) est atteint.

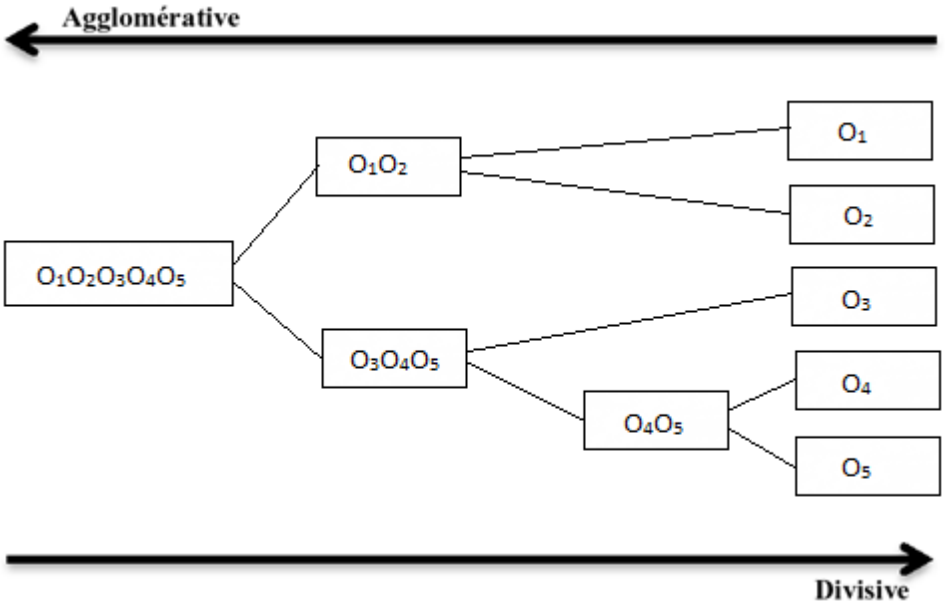


Figure 1.3 Exemple d'une classification hiérarchique.

Les avantages de la classification hiérarchique sont :

- Facilité pour traiter différentes formes de similarité ou de distance entre objets
- Applicable aux différents types d'attributs
- Une flexibilité en ce qui concerne le niveau de granularité

Les inconvéniants de la classification hiérarchique sont :

- Choix du critère d'arrêt qui reste vague
- Fait que la plupart des algorithmes hiérarchiques ne revoient pas les clusters intermédiaires qu'une fois qu'ils sont construits pour les améliorer.
- Les méthodes de classification hiérarchique basées sur des métriques de liens donnent comme résultats des clusters de formes convexes.
- La matrice de distance N × N, qui calcule les similarités ou dissimilarités entre les données est trop grande. Pour surmonter cette limite, plusieurs heuristiques sont possibles pour réduire la taille de cette matrice. Cela peut être réalisé en omettant des entrées inférieures à un certain seuil, ou en utilisant seulement un certain sous-

ensemble de représentants des données, ou en gardant pour chaque individu qu'un certain nombre des plus proches voisins [17].

1.4.2 La classification par partitionnement

La classification non supervisée par partitionnement consiste à trouver une partition de données, en assurant la similarité intra classe et la dissimilarité inter classe. Les techniques de clustering par partitionnement sont divisées en deux sous catégories principales [28]. La première approche, représente les algorithmes basés sur les centroïdes, où chaque cluster est représenté par son centre (le centre de cluster est calculé selon la distance entre tous ses objets), c'est-à-dire, le centroïde est un vecteur calculable, et il n'est pas un objet du cluster. La deuxième approche, représente les algorithmes basés sur les médoïdes, où chaque cluster est représenté par un objet représentatif, puis itérativement remplacer un par un autre si cela permet de réduire la distance globale entre les objets du cluster. un exemple sur la classification par partitionnement est présenté par la figure 1.4 ci-dessous[17]. .

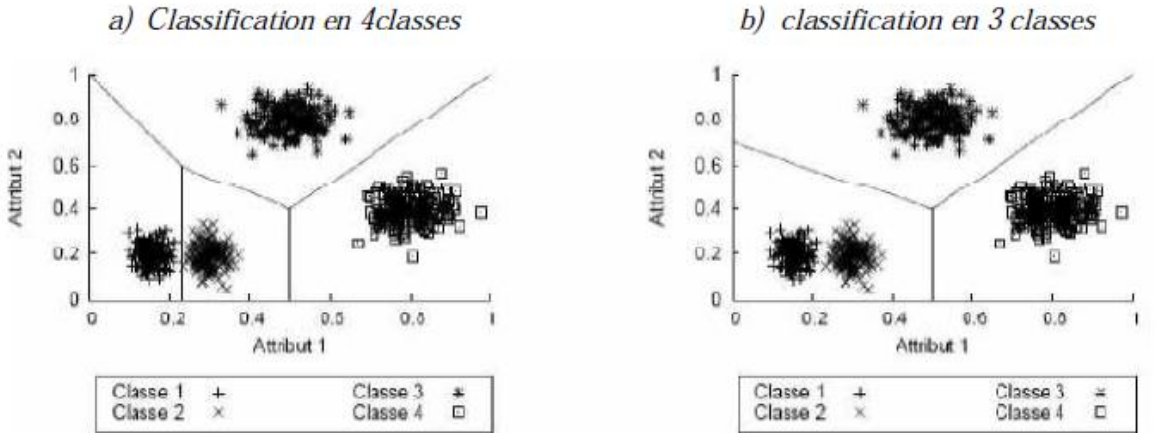


Figure 1.4 Classification par partitionnement

Dans ce que suite, nous allons décrire les deux algorithmes les plus connus, K_means et K_medoid [17].

1.4.2.1 La méthode K-moyenne

La technique de clustering de K_moyenne est très simple, La somme des dispersions (distances) entre un point et son centroïde, exprimée par une distance appropriée, est utilisée comme fonction objective. Par exemple, la fonction objective basée sur la somme des carrées des erreurs (en Anglais Sum Of Squared Errors (SSE)) entre les points xi et les centroïdes wj correspondant, est égale à la variance intra-cluster totale [05] .

$$E(C) = \sum_{j=1:K} \sum_{\mathbf{x}_i \in A} \|\mathbf{x}_i - \mathbf{w}_j\|^2$$

L'algorithme de base de K-moyennes est décrit comme suit :

Algorithme 1.1 K_moyenne
Entrées : k le nombre maximum de classes désiré. Début 1- Choisir k individus au hasard (comme centre des classes initiales 2- Affecter chaque individu au centre le plus proche 3- Recalculer le centre de chacune de ces classes 4- Répéter l'étape (2) et (3) jusqu'à stabilité des centres 5- Éditer la partition obtenue. Fin

Parmi les méthodes connues sur le principe de k-means nous citons les méthodes de centres mobiles .

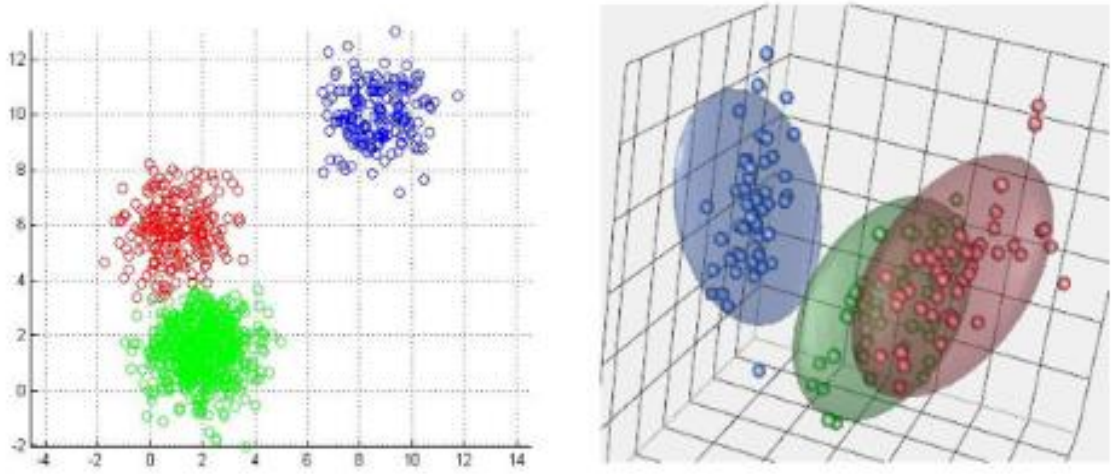


Figure 1.5: K_moyenne (gauche) a 2d. K- moyenne (droite) à 3d.

1.4.2.1.1 Les méthodes de centres mobiles :

consiste à réitérer les deux étapes suivantes :la premiere est de réaffecter tous les points à leurs centroïdes proches la deuxieme est de recalculer les centroïdes des nouveaux groupes formés.Les itérations continuent jusqu'à ce qu'un critère d'arrêt soit attein.l' algorithme de centres mobiles est presenté dans le tableau ci-dessous :

Algorithme1.2 : méthode de centres mobiles
Entrées : k le nombre maximum de classes désiré. Début 1-Choisir k individus au hasard (comme centre des classes initiales) 2-Affecter chaque individus au centre le plus proche ce qui donne une partition en k classes P1 = $\{c_i, \ldots, c_k\}$ 3-On calcule les centres de gravité des chacune des classes de P1 Ce qui donne k nouveaux centres de classes. 4-Répéter l'étape (2) et (3) jusqu'à deux itérations successives donnent la même partition 5-Éditer la partition obtenue. Fin

1.4.2.2 Les méthodes K-médoïdes

En statistiques, un médoïde est le représentant le plus central d'une classe. L'algorithme des k-medoids (k-médoïdes) est un algorithme de partitionnement plus robuste vis-à-vis les données aberrantes que celui des (k-moyennes). Comme les k-moyennes, l'algorithme des k-médoïdes minimise l'erreur quadratique moyenne qui est la distance entre les points de la classe et le point central (ou médoïde). Pour les k-moyennes, chaque cluster est représenté par son centre tandis que dans les k-médoïdes, chaque cluster est représenté par un des objets du cluster.

Dans les méthodes C − De_Ii_E5 une classe est représentée par un de ses points. C'est un algorithme efficace puisqu'il traite n'importe quel type d'attributs et que les medoïdes ont une résistance intégrée contre les données atypiques puisque les points périphériques des clusters ne sont pas pris en compte. Quand les medoïdes sont sélectionnés, les classes sont définies comme des sous-ensembles de points proches des medoïdes respectifs, et la fonction objective est définie comme la distance moyenne ou une autre mesure de dissimilarité entre un point et son medoïde [05].

1.4.3 Les méthodes à base de densité

Un ensemble ouvert dans l'espace Euclidien peut être divisé en un ensemble de ses composantes connectées. L'implémentation de cette idée a besoin de tels concepts comme la densité, la connectivité et les limites (extrémité, bordure, frontière) plutôt que la distance. Ils sont fortement reliés aux plus proches voisins d'un point. Un cluster, défini comme une composante densément connectée, évolue dans n'importe quelle direction où la densité le mène. Par conséquent, ces algorithmes sont capables de découvrir des clusters de formes arbitraires. Ainsi, ce principe fournit une protection naturelle contre les données aberrantes. Les clusters ne sont pas nécessairement convexes (entièrement contenus/remplis). Le bruit et les données isolées sont identifiés. Le nombre de classes n'a pas besoin d'être prédéfini préalablement. Un seul parcours de la base de données est effectué pour la réalisation de ces clusters [21] ; les clusters sont des régions denses séparées par des régions qui le sont moins son principe repose sur deux concepts :

- un point est voisin d'un autre point s'il est à une distance inférieure à une valeur fixée, cette valeur est à choisir selon l'échelle de l'étude et l'exploitation de la solution recherchée ;

- un point est dense si le nombre de ses voisins dépasse un certain seuil. Ce critère est aussi défini par l'utilisateur selon les objectifs de son étude [30].

Il existe deux types d'approches à base de densité. Le premier type d'approche attache la densité a un point qui appartient aux données d'apprentissage. Il compte des algorithmes spécifiques comme DBSCAN, GDBSCAN, OPTICS et DBCLASD. Le deuxième type d'approche attache la densité à un point dans l'espace d'attributs. Il inclut l'algorithme Denclue.

1.4.4 Les méthodes à base de grille

Dans la section précédente, nous avons utilisé des concepts majeurs tels que la densité, les limites et la connectivité. Une autre manière de traiter ces concepts est d'extraire la topologie de l'espace des attributs sous-jacent. Pour limiter les combinaisons (possibilités) des recherches, on considère des segments multi-rectangulaires. On rappelle qu'un segment (appelé aussi *cube, cellule, région*) est un produit cartésien direct des sous-rangs de chaque variable (adjacent dans le cas des variables numériques). Le segment élémentaire qui correspond à des sous-rangs d'une seule valeur ou un seul histogramme est appelé unité. L'idée de ces méthodes est qu'on divise l'espace de données en un nombre fini de cellules formant une grille. Ce type d'algorithme est conçu pour des données spatiales [30].

En fait, avec une telle représentation des données, au lieu de faire la classification dans l'espace de données, on le fait dans l'espace spatial en utilisant des informations statistiques des points dans la cellule. Les méthodes de ce type sont hiérarchiques ou de partitionnement parmi ces méthode nous citons :STING, CLIQUE et WaveCluster.

Le partitionnement des données est induit par l'appartenance des points dans des segments qui résultent de l'espace de partitionnement, alors que l'espace de partitionnement est basé sur les caractéristiques de la grille extraites des données d'entrée.

Un avantage de ce traitement indirect est que l'accumulation des données sous forme de grille rend les techniques à base de grille indépendante par rapport à l'ordre (rangement) des données. En revanche, les méthodes par réaffectation et les méthodes incrémentales sont très sensibles à l'ordre des données. Tandis que les méthodes à base de densité fonctionnent mieux sur des variables numériques, les méthodes à base de grille fonctionnent avec des variables de différents types [30].

1.5 Conclusion

Dans ce chapitre, nous avons présenté les principes fondamentaux de la classification non suppervisée de données. En deuxieme temps, nous avons passé en revue sur quelques méthodes classiques utilisées pour réaliser cette tache avec une concentration sur leurs principes et leurs avantages.

CHAPITRE 2

METHODES D'OPTIMISATION POUR LA RESOLUTION DES PROBLEMES DIFFICILES

2.1 Introduction

Les problèmes de classification des données sont des problèmes d'optimisation NP-complet. Nous irons voir par la suite qu'il existe des problèmes qui ont des complexités différentes parmi eux ,qui appartiennent à la classe NP, l'existence d'algorithmes polynomiaux semble peu réaliste. Ainsi, différentes méthodes de résolution (méthodes exactes ou heuristiques) sont largement utilisées pour appréhender les problèmes NP-difficiles. Nous exposons dans ce chapitre les grandes lignes des méthodes d'optimisation les plus connues dans la littérature[03].

2.2.Optimisation combinatoire

un modèle d'un problème d'optimisation combinatoire P=(S,Ω,f) telque S est un espace de recherche défini au-dessus d'un ensemble fini de variables de décision discrètes.

Ω : est un ensemble de contraintes sur les variables.

$f: S \rightarrow \mathbb{R}^+$ est une fonction à minimiser appelée fonction objectif .

L'ensemble des solutions réalisables S_Ω est l'ensemble des éléments de S qui vérifient toutes les contraintes de Ω . Un élément $S^* \in S_\Omega$ est un optimal global si et seulement si $\forall\, s \in S_\Omega$,$f(s^*) \leq f(s)$.L'ensemble de solutions globalement optimales est dénoté $s^* \subseteq S$, et pour résoudre un problème d'optimisation combinatoire on doit trouver une solution $s^* \in S^*$ [03].

2.2.1 Definition :Un problème d'optimisation est défini par une fonction objectif F qu'on cherche à minimiser (ou maximiser), et un ensemble de contraintes que doivent respecter les solutions de ce problème. Un problème d'optimisation sera défini par :

- Un ensemble d'instances I.
- Une fonction S telle que pour tout $x \in I, S(x)$ représente l'ensemble des solutions réalisables pour X.
- Une fonction m à valeur entière définie sur tous les $x \in I$ et $y \in S(x)$. Cette mesure m représente la fonction objectif d'une solution Y pour l'instance X .
- Un objectif $f \in \{\min, \max\}$ précisant si Ⅱ est un problème de minimisation ou de maximisation.A toute instance X, on associe une solution et la valure $m^*(x)$ déffinissant un optimum [03].

2.2.2 Théorie de la Complexité des Algorithmes

La théorie de la complexité des algorithmes a donné un sens précis au terme d'algorithme efficace et de problème difficile, Alors évaluer la complexité d'un algorithme consiste à trouver en fonction de la taille des données, le nombre d'opérations élémentaires nécessite par cet algorithme. Alors on dit qu'un algorithme est efficace si le nombre des opérations nécessaires pour résoudre un problème est donné par une fonction polynomiale d'un paramètre caractérisant la taille du problème considéré [03].

2.3 Les Classes des Problèmes

2.3.1 La Classe P

Un problème de décision est dans la classe P s'il peut être décidé sur une machine déterministe en temps polynomial par rapport à la taille de la donnée. On qualifie alors le problème de polynomial, c'est un problème de complexité O(x^k) pour un certain k [20].

2.3.2La Classe NP

C'est la classe des problèmes qui peuvent être décidés sur une machine non déterministe en temps polynomial. De façon équivalente, c'est la classe des problèmes qui admettent un algorithme polynomial capable de tester la validité d'une solution du problème. On distingue deux grandes sous-classes :

- La classe P : est la classe des problèmes résolubles par un algorithme polynomial déterministe. C'est la classe des problèmes les plus faciles de NP [03].
- La classe NP-complet : cette classe est basée sur la notion de réduction polynomiale [02].

2.4 Classification des Méthodes d'Optimisation

Plusieurs classifications ont vu le jour, elles sont représentées dans la figure 2.1 .

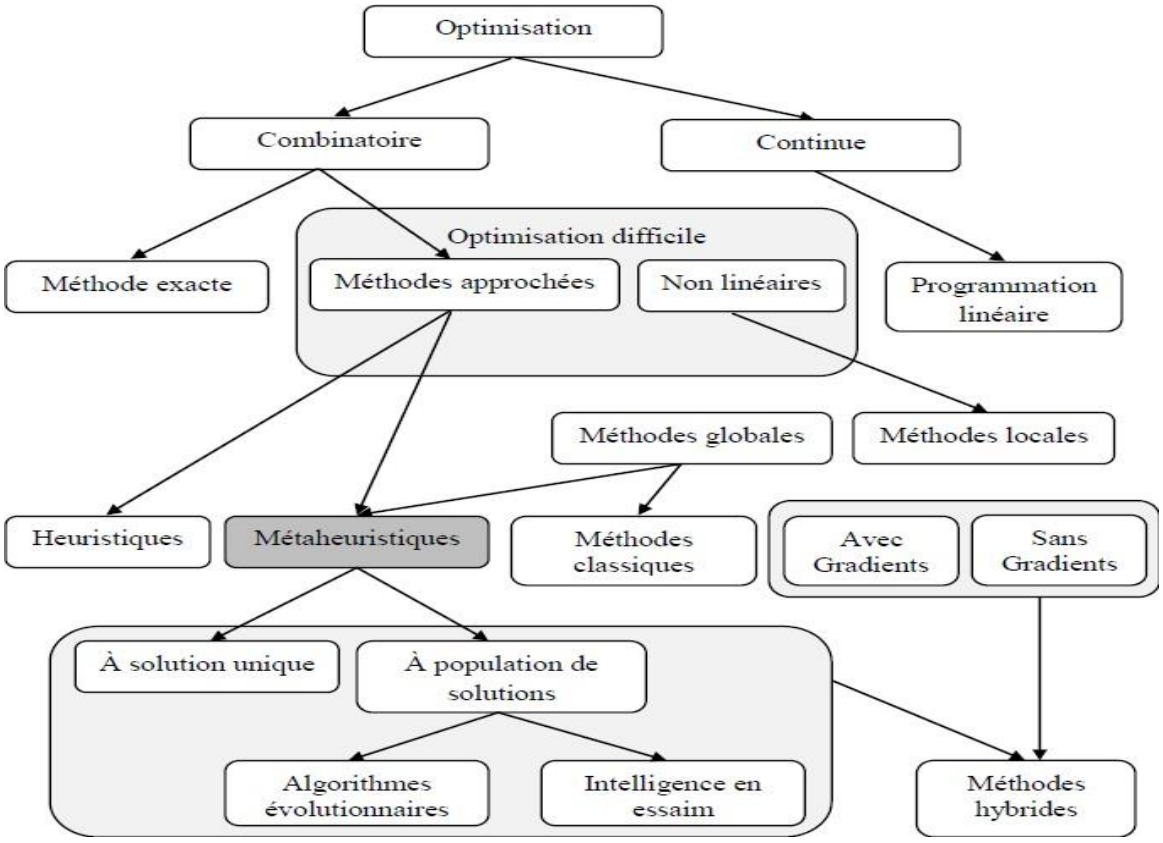


Figure2.1 Classification des différents types de méthodes d'optimisation [29]

2.4.1 Les Méthodes Exactes

Elles sont basées sur une recherche exhaustive et en évaluant chaque solution de , jusqu'à trouver la solution optimale globale, comme : Brunch and Bound et la programmation dynamique. Les méthodes de cette classe sont vite devenues inutilisables pour des instances de grandes tailles[16].

2.4.2 Les Métaheuristiques : ce sont des algorithmes qui guident et modifie d'autres heuristiques pour produire des solutions [22]. Les métaheuristiques utilisent une ou plusieurs informations sur le problème qui changent durant le processus de recherche. Elles se subdivisent en deux sous-classes : Les méthodes a une solution et les méthodes a population.

2.5 Quelques méthodes a une solution :

2.5.1 La méthode de descente

Elle consiste à rechercher dans le voisinage de la solution courante, une solution de coût plus faible. Définition: Soit X l'espace de recherche d'un problème d'optimisation et soit f la fonction objectif. Une fonction de voisinage N est une association $x \rightarrow 2^x$, définissant,

pour chaque point s ∈ X , Une solution $\acute{s} \in N(s)$ tel que *N(s)* ⊆ X est une voisine de *s*. Une solution s ∈ X est un minimum local relativement à la structure de voisinage N si *f(s)*≤ $f(\acute{s})$ $\forall \acute{s} \in N(s)$.Une solution s ∈X est un minimum global si *f(s)*≤ $f(\acute{s})$ $\forall \acute{s} \in X$ [33]. L'algorithme 3.1 présente un pseudo code pour la méthode de descente simple.

Algorithme 2.1 : Descente Simple
1: initialisation : trouver une solution initiale x ;
2: répéter
3: recherche de voisinage : trouver une solution ẋ ∈ N(x) ;
4: si f(ẋ) < f(x) alors ẋ ≔ x ;
6: fin si ;
7: jusqu'à f(y) ≥ f(x), ∀y ∈ N(x).

2.5.2 Le recuit simulé

Cette méthode est issue d'une analogie entre le phénomène physique de refroidissement d'un corps en fusion, qui le conduit à un état solide. L'analogie exploitée par le recuit simulé consiste à considérer une fonction à minimiser comme fonction d'énergie, et une solution peut être vue comme un état donné de la matière dont est son énergie [33].

Algorithme 2.2 : Recuit simulé
1: initialisation :
Trouver une solution initiale x ;
Poser une température initiale t ;
2: répéter
3: recherche de voisinage : trouver une solution ẋ ∈ N(x) ;
4: déterminer Δc = f(ẋ) − f(x) ;
5: obtenir r ∈ [0.1] ;
6: si Δc < 0 ou $e^{(-\Delta c/T)}$ > r alors ẋ ≔ x ;
8: fin si ;
9: réduire la température t selon un schéma de refroidissement ;
10: jusqu'à un critère d'arrêt satisfait ;

2.5.3 La recherche tabou

La métaheuristique Recherche Tabou développée par F. Glover en 1990 [06] permet de contrôler le problème des optimums locaux. L'idée consiste à garder la trace du cheminement passé dans une mémoire et de s'y référer pour guider la recherche.

2.5.3.1 Principe de base

Le principe de cette méthode est simple, en partant d'une solution initiale x∈ X qui progresse itérativement à travers l'espace de recherche, en se déplaçant vers une solution située dans le voisinage , et en choisissant toujours la meilleure solution voisine même si cette dernière est moins bonne que la solution courante, ceci pour éviter de se bloquer dans un optimum local. Mais il induit un risque de tomber dans un cycle. En effet, lorsque l'algorithme a quitté un optimum local quelconque par acceptation de la dégradation de la fonction objectif, il peut revenir sur ses pas, à l'itération suivante .

2.5.3.2 La liste Tabou **:**La liste tabou est généralement gérée comme une liste circulaire en remplaçant toujours l'élément tabou le plus ancien**[**06**]**.

Algorithme 2.3 : Recherche Tabou
1 : Initialisation X :=solution initial ; $f_{min} \coloneqq f(x)$; $x_{min} \coloneqq x$; Tabou:=liste de solutions x longueur L; Tabou *:= Vide ;* 2 : répéter *3:* générer un voisinage $N(x) telque\ x_i \in N(x\ et\ x_i \notin tabou)$; *4 :* $f(\acute{x}) = min_{1<i<N}[f(x_i)]$**;** *5 :* Ajouter $(\acute{x}, \acute{t}ablou)$**;** *6 : x :=* $\acute{x}$*;* *7 : si* $f(x) < f_{min}$ *alors* *8 :* $f_{min:=f(x)}$ *;* *9 :* $x_{min} \coloneqq x$*;* *10: fin si ;* *11: jusqu'à conditions d'arrêt satisfaites ;*

2.6 Quelques Méthodes à population

Ces méthodes manipulent un groupe de solutions admissibles à chacune des étapes du processus de recherche. L'idée centrale consiste à utiliser régulièrement les propriétés collectives d'un ensemble de solutions qu'on peut distinguer, appelé population dans le but de guider efficacement la recherche vers de bonnes solutions dans l'espace X. Parmi ces méthodes on peut citer les algorithmes génétiques(GA), l'optimisation par essaim de particules (PSO) [13].

2.6.1 Optimisation par Essaim Particulaires

L'optimisation par essaims particulaires est une méta heuristique d'optimisation, proposée par Russel Ebenhart et James Kennedy en 1995 . Il s'agit d'une méthode faisant appel à une population d'agents, appelés ici particules , dans laquelle, à chaque pas de temps, les meilleurs (selon un critère prédéfini) sont plus ou moins imités par les autres.

Au départ de l'algorithme, un essaim de particules est réparti au hasard dans l'espace de recherche, chaque particule ayant une vitesse aléatoire et se déplace dans l'espace de recherche. La position de chaque particule est représentée par un vecteur de dimension D (D est le nombre de variables pour chaque objet) dans l'espace du problème $X_i = (x_{i1}, x_{i2}, \ldots . x_{iD})$, i = 1,2,…,N talque (N c'est la taille de la population) et sa vitesse $V_i = (v_{i1}, v_{i2}, \ldots . v_{iD})$ est définie comme le changement de sa position. Les performances de l'algorithme sont évaluées par une fonction fitness prédéfinie à chaque itération de l'algorithme .Notons que :

- chaque particule est capable d'évaluer la qualité de sa position et de garder en mémoire sa meilleure performance.
- Chaque particule est capable d'interroger un certain nombre de ses congénères (ses informatrices, dont elle-même) et d'obtenir de chacune d'entre elles sa propre meilleure performance (la qualité afférente).
- A chaque pas de temps, chaque particule choisit la meilleure performance dont elle a connaissance, modifie sa vitesse en fonction de cette information et de ses propres données et se déplace en conséquence.
- Les équations du mouvement d'une particule

Après avoir trouvé les deux meilleures valeurs, les particules évoluent en mettant à jour leurs vitesses et positions selon les équations suivantes:

$$v_i(t+1) = w * v_i(t) + c_1 * r_1 * \left(p_{i_{best}} - x_i(t)\right) + c_2 * r_2 * \left(g_{best} - x_i(t)\right) \ldots . (1)$$

$$x_i(t+1) = x_i(t) + v_i(t+1) \ldots . (2)$$

Où , Et est la taille de l'essaim, est la particule qui a atteint la meilleure solution, : est la meilleure solution globale dans l'essaim, et sont des paramètres cognitifs et sociaux qui sont bornées entre 0 et 2, et sont deux nombres aléatoires, avec une distribution uniforme [0,1], : est un facteur utilisé pour contrôler l'équilibre de l'algorithme de recherche entre l'exploration et l'exploitation

Algorithme2.4 : PSO
t← 0;
Pour chaque particule faire
Initialiser sa position et sa vitesse ;
Initialiser $p_{i-best}(t)$;
Fin pour ;
Tant que critère d'arrêt n'est pas atteint faire
Choisir la particule $g_{best}(t)$ ayant la meilleure fitness ;
Pour chaque particule i faire
Calculer la vitesse $v_i(t+1)$ par l'équation (1) ;
Mettre à jour de position $x_i(t+1)$ selon l'équation(2) ;
Calculer la valeur de la fitness $f(x_i(t))$;
Si $f\left(x_i(t)\right) > f(p_{i-best}(t))$ alors f(p_{i-best})← $x_i(t)$;
Fin si ;
Fin pour ;
t← $t+1$;
Fin tant que ;

2.6.2 Optimisation par algorithme génétique (AG)

2.6.2.1 Origine: Les algorithmes génétiques fonctionnent sur une analogie avec la reproduction des êtres vivants. L'évolution des solutions en même temps comme dans l'évolution naturelle en biologie. Un AG a quatre composants : une population d'individus, où chaque individu de la population représente une solution candidate au problème d'optimisation, une fonction de fitness qui est une fonction d'évaluation par laquelle nous pouvons dire si un individu est une bonne solution ou non, une fonction

de sélection qui décide comment choisir les bons individus de la population actuelle pour la création de la génération suivante; et des opérateurs génétiques tels que le croisement et la mutation, qui explorent de nouvelles régions de l'espace de recherche tout en gardant une partie de l'information actuelle [12] .

2.6.2.2 Les composants d'algorithme génétique

2.6.2.2.1 Codage de données

Chaque paramètre d'une solution est assimilé à un gène, toutes les valeurs qu'il peut prendre sont les allèles de ce gène, on doit trouver une manière de coder chaque allèle différent de façon unique (établir une bijection entre l'allèle "réel" et sa représentation codée).

2.6.2.2.2 Génération de la population initiale

C'est lors de l'initialisation de l'algorithme génétique que les individus originaux sont générés. Un chromosome est une suite de gêne, on peut par exemple choisir de regrouper les paramètres similaires dans un même chromosome (chromosome à un seul brin) et chaque gène sera repérable par sa position. Chaque individu est représenté par un ensemble de chromosomes, et une population est un ensemble d'individus [12].

2.6.2.2.3 Sélection : La sélection permet d'identifier les individus susceptibles d'être croisés dans une population. Il existe plusieurs techniques de sélection. Nous présentons ici les quatre les plus utilisées parmi elles [12] :

- Sélection par rang : consiste à ranger les individus de la population dans un ordre croissant ou décroissant, selon l'objectif (fonction fitness).
- Sélection par roulette : elle consiste à créer une roue de loterie biaisée pour laquelle chaque individu de la population occupe une sélection de la roue proportionnelle à sa valeur d'évaluation.
- sélection aléatoire : cette sélection se fait aléatoirement, uniformément et sans intervention de la valeur d'adaptation. Chaque individu a donc une probabilité uniforme $\frac{1}{P_{size}}$d'être sélectionné, où p_{size} est le nombre total d'individus dans la population.
- sélection par tournoi : le tournoi le plus simple consiste à choisir aléatoirement un nombre k d'individus dans la population et à sélectionner celui qui a la meilleure performance. Les individus qui participent à un tournoi sont remis ou sont retirés

de la population, selon le choix de l'utilisateur. Avec le tournoi binaire, sur deux individus en compétition, le meilleur gagne avec une probabilité $p \in [0,5;\ 1]$

2.6.2.2.4 Croisement

L'opérateur de croisement combine les caractéristiques d'un ensemble d'individus parents (généralement deux) préalablement sélectionnés, et génère de nouveaux individus enfants. Là encore, il existe de nombreux opérateurs de croisement [13].

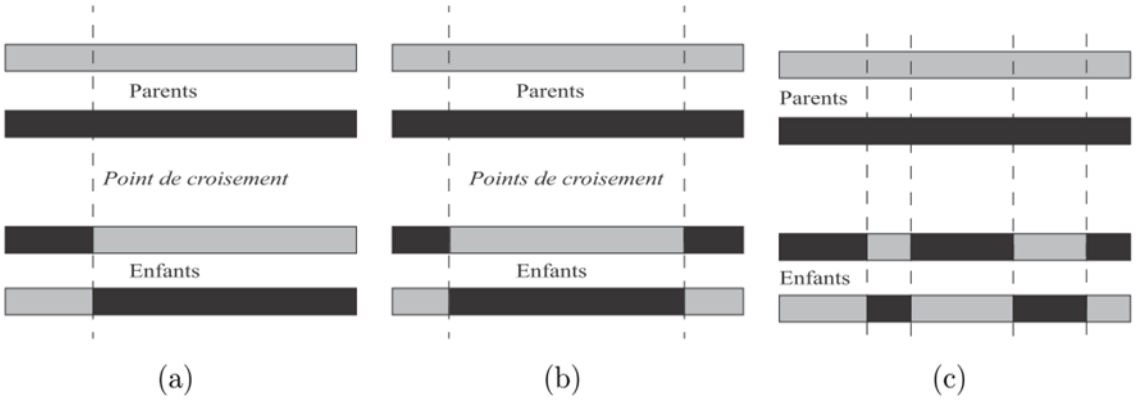


Figure 2.2 : Croisement : (a) croisement simple en un point, (b) croisement en deux points, (c) croisement uniforme.

2.6.2.2.5 Mutation

De façon aléatoire, un gène peut, au sein d'un chromosome être substitué à un autre. De la même manière que pour les enjambements, on définit ici un taux de mutation lors des changements de population qui est généralement compris entre 0,001 et 0,01. Il est nécessaire de choisir pour ce taux une valeur relativement faible de manière à ne pas tomber dans une recherche aléatoire et conserver le principe de sélection et d'évolution. La mutation sert à éviter une convergence prématurée de l'algorithme. Par exemple lors d'une recherche d'extremum la mutation sert à éviter la convergence vers un extremum local.

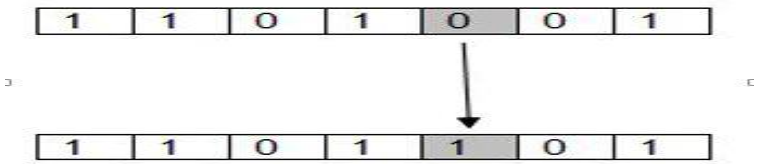


Figure 2.3 : une mutation dans le cas d'un codage binaire.

L'efficacité des algorithmes génétiques dépend fortement du réglage des différents paramètres caractérisant ces algorithmes, et qui sont parfois difficiles à déterminer. Des paramètres comme la taille de la population, le nombre maximal des générations, la probabilité de mutation pm, et la probabilité de croisement pc. Les deux premiers paramètres dépendent directement de la nature du problème et de sa complexité, et leurs choix doit représenter un compromis entre la qualité des solutions et le temps d'exécution. La probabilité de croisement pc est liée à la forme de la fonction d'évaluation. Son choix est en général heuristique. Plus sa valeur est élevée, plus la population subit des changements importants. La probabilité de mutation pm est généralement faible puisqu'un taux élevé risque de conduire vers un optimum local[12].

Algorithme 2.5 : Algorithme génétique
1: initialisation: générer une population initiale P de solutions de taille |P|= n ; 2 :Evaluer P ; répéter 3: sélection : choisir 2 solutions x et x' par une technique de Sélection ; 4: croisement : combiner les solutions parents x et x' pour former une solution enfant y ; 5: mutation de y ; 6: choisir une solution individuelle y' pour être remplacée dans P ; 7: remplacer y' par y dans P ; 8 :Evaluer P ; jusqu'à critère d'arrêt satisfait ;

2.7 Conclusion:

L'optimisation combinatoire offre un ensemble de méthodes de résolution (exactes et approchées) qui permet de résoudre les problèmes difficiles en un temps réduit et raisonnable, avec une bonne qualité de solution. Nous avons cité un panorama de métaheuristiques utilisées pour la classification non supervisée. Cependant, ces métaheuristiques sont proposées pour résoudre d'autres problèmes, puis, elles sont appliquées à la classification non supervisée. Ce qui pose le problème d'adaptation des paramètres avec le problème de clustering.

CHAPITRE 3

MÉTHODE PROPOSÉE

3.1Introduction

Dans ce chapitre, on va présenter deux métaheuristiques à population : L'algorithme à base de biogéographie BBO ($BBO: Biogeography - Based\ Optimization$) et les algorithmes génétiques GA ($GA: Genetic\ algorithm$).

3.2 Optimisation par algorithme BBO

3.2.1 Origine de BBO

L'algorithme à base de biogéographie BBO a été développé par Dan Simon en 2008 [04] elle trouve ses origines dans la théorie de l'équilibre dynamique (appelée aussi théorie de la biogéographie insulaire). La théorie de la biogéographie consiste en l'étude de la répartition spatiale des espèces vivantes (végétales et animales) et des causes de cette répartition. Elle traite de la façon dont la richesse en espèces (nombre d'espèces) est maintenue dans un système d'île 7 qui sont sujettes à l'immigration et sur lesquelles des espèces s'éteignent . Elle stipule que les milieux insulaires sont à l'origine vides d'espèces et que celles-ci y arrivent peu à peu en provenance de régions vastes (désignées sous le terme de « continents » bien qu'il ne s'agisse pas forcément de continents à proprement parler) ou d'îles voisines. Certaines espèces sont d'ailleurs mieux outillées que d'autres pour conquérir de nouveaux territoires, elles ont donc des capacités de colonisation des milieux insulaires plus grandes que d'autres. Les interactions compétitives sur l'île tendent par contre à accélérer les extinctions . Le croisement de ces deux processus dynamique permet d'expliquer la richesse actuelle du peuplement. A l'équilibre, il y a un remplacement constant des espèces[04].

3.2.2 Principe de L'algorithme BBO

L'algorithme BBO s'inspire de la théorie de la biogéographie insulaire. Les deux principaux opérateurs qui gouvernent son fonctionnement sont la migration et la mutation. L'algorithme commence par générer un nombre fini d'individus choisis généralement par tirage aléatoire uniforme dans l'espace de recherche formant la population initiale. Après évaluation de la population initiale, certains individus sont choisis pour participer à l'opération de migration qui permet de créer un nouvel ensemble d'individus. Les descendants vont être à leur tour mutés. Le taux de mutation fixe la proportion de la population qui sera renouvelée à chaque génération. Enfin, une phase de remplacement consiste à remplacer les parents par les nouveaux descendants, afin de former une nouvelle population, de la même taille qu'au début de l'itération. L'algorithme se termine après un certain nombre de générations [31].

3.2.3 Composants d'algorithme BBO

3.2.3.1 Habitat

L'algorithme BBO manipule une population d'individus appelés îles (ou habitats). Chaque île représente une solution possible au problème à résoudre. La « fitness » de chaque île est déterminée par son HSI (Habitat Suitability Index), une mesure de la qualité d'une solution candidate, et chaque île est représentée par des $SIVs$ (Suitability Index Variables). Une bonne solution au problème d'optimisation est une île avec un grand nombre d'espèces, ce qui correspond à une île avec un faible HSI [31].

3.2.3.2 Opérateur de migration et operateur d'émigration

L'opérateur de migration consiste à remplacer les valeurs SIV d'un habitat H_i par des valeurs SIV d'un autre habitat Hj, il est similaire à l'opérateur de croisement dans un algorithme génétique. Alors que l'opérateur d'émigration est appliqué au niveau de chaque habitat H_i de la population, il consiste à remplacer la valeur SIV d'un habitat Hi par une valeur SIV choisi aléatoirement de l'espace de solutions. L'opérateur d'émigration est similaire de l'opérateur de mutation dans un algorithme génétique. Ces deux processus sont présentés dans la figure ci-dessous [31].

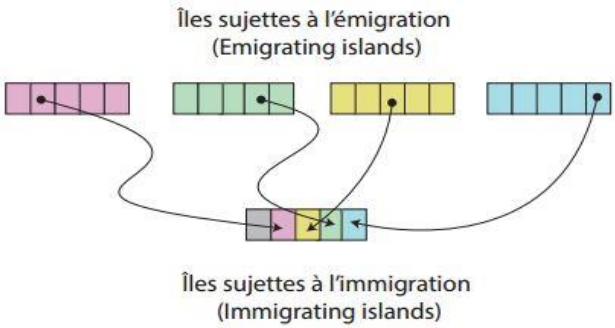


Figure 3.2 : Opérateur de migration

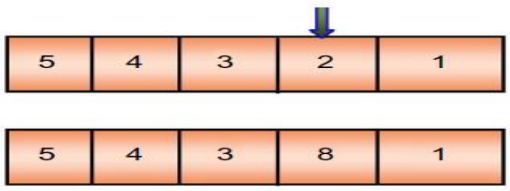


Figure 3.3 : Opérateur d'émigration

Selon la théorie de Mac Arthur et Wilson [24], Dans BBO, chaque habitat a son propre taux d'immigration (λ) - arrivées venant de l'extérieur - et son taux d'émigration (μ) - départs vers l'extérieur. Ces paramètres dépendent du nombre d'espèces (S) présentes sur l'île. L)e taux d'immigration (λ) décroît avec l'augmentation du nombre d'espèces (S) déjà présentes sur l'île. Plus le nombre d'espèces déjà installées sur l'île augmente, de moins en moins d'immigrants appartenant à une nouvelle espèce rejoignent l'île. Mais, au fur et à mesure que le nombre d'espèces déjà présentes sur l'île diminue, plus le taux d'immigration n'augmente. Le taux d'immigration maximale (I) est atteint lorsque l'île est vide. Une fois que toutes les espèces sont présentes sur l'île, alors $S = S_{max}$ (capacité maximale de l'île) et le taux d'immigration tombe à zéro, ne favorisant plus l'installation de nouveaux arrivants (plus l'île est peuplée, moins les espèces étrangères ont de chances de s'y implanter). Le taux d'immigration, quand il y a S espèces sur l'île, est donné par :

$$\lambda_S = I\left(1 - \frac{S}{S_{max}}\right)$$

Le taux d'émigration (μ) augmente avec le nombre d'espèces (S) présentes sur l'île. Le taux d'émigration maximum (E) se produit lorsque toutes les espèces sont présentes sur l'île ($S = S_{max}$), et devient nul si les espèces présentes sur l'île s'éteignent (ou quittent 'île). Le taux d'émigration quand il y a S espèces sur l'île est donné par[31]. :

$$\mu_{S=} E\left(\frac{S}{S_{max}}\right)$$

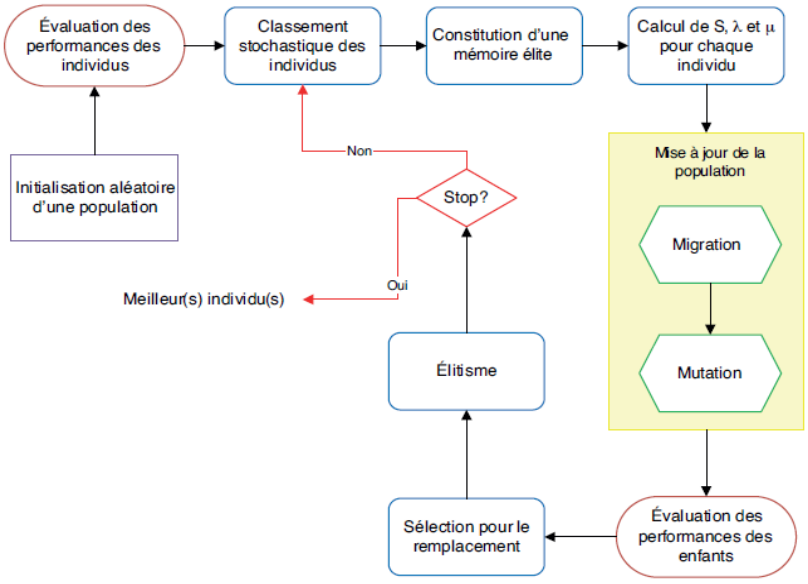


Figure 3.4 : Les étapes de l'algorithme BBO

3.2.4 Adaptation de l'algorithme BBO au problème de clustering

Après la présentation de l'algorithme BBO et ses composants essentiels, dans cette section on va adapter cet algorithme pour trouver une solution approchée au problème de clustering . Afin d'illustrer comment appliquer l'algorithme pour trouver la classificatin optimale, nous pouvons nous servir du jeu de données Fleurs d'iris bien connu. Ce jeu de données comprend un total de 150 observations, réparties de manière égale entre les trois espèces de fleurs d'iris (setosa,virginica et versicolor). Quatre caractéristiques sont mesurées pour chaque observation (c.-à-d. la longueur et la largeur du sépale et du pétale, en centimètres).

3.2.4.1 Le codage proposé

Dans un algorithme BBO, un habitat représente une solution au problème de clustering peut être représentée par une table de partitionnement ou un ensemble de centroïdes de clusters.Pour générer notre problème,on presente chaque habitat par une séquence de nombres réels représentant les K centres de clusters. Pour un espace à N dimensions, la longueur d'un habitat est constitué de N * K sivs , où les premières N positions (siv) représentent les N dimensions du premier centre de cluster, les N positions suivantes représentent celles du deuxième groupe centre, et ainsi de suite [13] un habitat est créé comme en figure 3 .

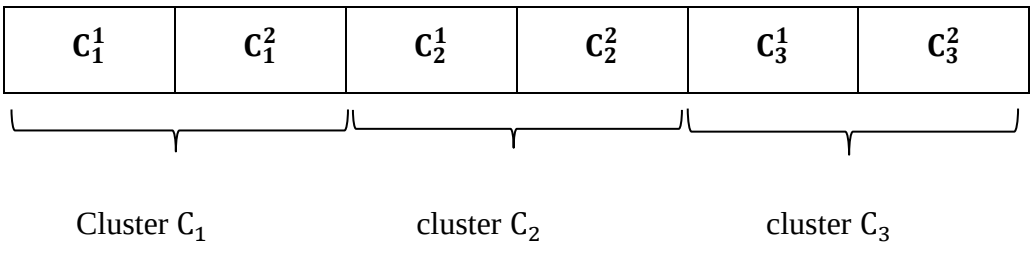


Figure 3.5: Encodage d'un Habitat

3.2.4.2 Fonction objective proposée **:** HSI (habitat suitibilité index)

Déterminer le nombre optimal de classes dans un jeu de données est l'un des aspects les plus difficiles de la processus de regroupement . La plupart des approches trouvées dans la littérature déterminent le bon nombre de classes selon des critères basés sur des classes données. Cependant, le BBO n'optimise pas uniquement les regroupements pour un nombre donné de classes mais aussi le nombre de classes, c'est-à-dire l'objectif fonction doit maximiser l'homogénéité au sein de chaque groupe, l'hétérogénéité entre les groupes, ainsi que doit être le plus élevé lorsque la solution représente le bon nombre de clusters [13].

Dans notre etude on a fixé le nombre de classes dés le début et on cherche a minimiser la distance entre les point et les centres de classes a qui ces points appartiennent pour aborder cet objectif on va utiliser la fonction qui assigne chaque point a la classe la plus proche puis calcule la somme des distances entre les points de meme classe et les centres de ces classes en utilisant la matrice de distance , on a nommé cette fonction " WCDS " within-cluster distance sum" [19].

```
fonction [Z] = ClusteringCost(H, X)
Debut
MatDist = pdist2(X, H) ;            % Calculater la matrice distance
 [dmin] = min(MatDist, [], 2) ;     % trouver les distances minimales
WCD = sum(dmin);                    % Somme de min distances.
 Z=WCD;
Fin
```

Tableau 3.1 : Fonction d'évaluation

3.2.4.3 Initialisation des paramètres :

Paramètres	Notation	Valeur
Taille de la population	nPop	50
Paramètre d'élitisme	nKeep	2
Taux maximum d'immigration	I	1
Taux maximum d'émigration	E	1
Nombre de générations	Maxi	200
Domaine de définition des variables	[varmin, varmax]	[0, 255]
Dimension du problème	nVar	4
Constante de mutation	pMutation	0,3

Tableau 3.2: Valeurs des paramètres de l'algorithme BBO [04]

3.2.4.4 Population initiale

dans notre travail an a créer une population initiale par génération aléatoire (Random) d'un nombre d'individus (habitats) ; chaque habitat est initialisé aléatoirement par k centres de classe alors une population est comme suit:

Pop=
$$\begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} & \cdots & x_{1m} \\ x_{21} & x_{22} & x_{23} & x_{24} & \cdots & x_{2m} \\ x_{31} & x_{32} & x_{33} & x_{34} & \cdots & x_{3m} \\ . & . & . & . & & . \\ . & . & . & . & & . \\ . & . & . & . & & . \\ x_{N1} & x_{N2} & x_{N3} & x_{N4} & \cdots & x_{Nm} \end{bmatrix}$$

Chaque ligne représente un habitat

3.2.4.5 Évaluation :

Pour chaque habitat on calcule la fonction objective ou bien "HSI" , on fait appelle à la fonction *ClusteringCost* citée dans la section **3.2.3.2**.Après cette évaluation nous devons trier les habitats selon l'ordre croissant de leurs *HSI* en utilisant la fonction *Sort*.

3.2.3.6 Sélection pour exécuter cette étape on a besoin de calculer le "immigration rate "λ" et le "émigration rate "μ" pour toute las habitats de la population telque :

$$\lambda = \mathrm{I}\left(1 - \frac{\mathrm{S}}{\mathrm{S}_{\max}}\right), \qquad \mu = \mathrm{E}\left(\frac{\mathrm{S}}{\mathrm{S}_{\max}}\right)$$

Après le calcul des λ et μ on fait appel à l'opérateur de migration puis l'operateur d'émigration.

3.2.4.6 Opérateur de migration

L'idée de la migration est de créer des descendants à partir de plusieurs individus de la population. Le nombre et la destination des SIVs migrants est décidé aléatoirement en fonction des taux d'émigration et d'immigration, comme le montre l'algorithme 3.1.

3.2.4.7 Operateur d'émigration : le HSI d'habitat peut changer soudainement à cause d'événements apparemment fortuits. Nous modélisons cela en BBO en tant que mutation du SIV, et nous utilisons les probabilités de dénombrement des espèces pour déterminer les taux de mutation voir tableau 3.2[04].

3.2.4.8 Élitisme

La stratégie élitiste consiste à conserver le(s) meilleur(s) individu(s) à chaque génération. Ainsi l'élitisme empêche l'individu le plus performant de disparaître au cours du remplacement ou que ses bonnes combinaisons soient affectées par les opérateurs de variation. Après chaque évaluation de la performance des individus à une génération g donnée, les n élit meilleurs individus de la génération précédente (g − 1) sont réintroduits dans la population si aucun des individus de la génération g n'est meilleur qu'eux [16].

L'amélioration des habitats continue à chaque itération jusqu'à atteindre le critère d'arrêt et obtenir la solution optimale .

3.2.5 Algorithme BBO

Le processus de l'algorithme BBO pour résoudre les problèmes de "clustering" peut être résumé comme dans le tableau suivant:

```
1 Générer aléatoirement un ensemble de solutions initiales (îles)
2 tant que le critère d'arrêt n'est pas atteint faire
3 |     Évaluer la fitness (HSI) de chaque solution
3 |   Calculer le nombre d'espèce S, le taux d'immigration λi et d'émigration μipour
4 |   chaque solution
5 |   Migration :
6 |   pour i = 1 à N faire
7 |   |   Utiliser λi  pour décider, de manière probabiliste, d'immigrer à Xi
8 |   |   si rand(0, 1) < λi alors
9 |   |   |   pour j = 1 à N faire
10|   |   |   |   Sélectionner l'île d'émigration Xj avec une probabilité ∝μj
11|   |   |   |   si rand(0, 1) < μj alors
12|   |   |   |   |   Remplacer une variable (SIV ) choisie aléatoirement dans Xipar
  |   |   |   |   |   La variable SIV correspondante dans Xj
13|   |   |   |   fin
14|   |   |   fin
15|   |   fin
16|   fin
17|   Émigration: muter les habitats  au taux de mutation
  |   si rand(0, 1) < pMutation alors
  |   Remplacer une variable (SIV ) choisie aléatoirement dans Xipar une valeur
  |   δ ∈ (minval , maxval)
18|   Remplacement de la population par les descendants
19|   Implémenter l'élitisme
20 fin

21 retourner la meilleure solution trouvée
```

Tableau 3.3 Algorithme BBO [13].

3.3 Adaptation d'algorithme génétique au problème de clustering

On a déjà présenté l'algorithme génétique et ses composants essentiels dans le chapitre précédent, dans ce chapitre on va adapter cet algorithme pour trouver une solution optimale au problème de clustering. L'algorithme proposé consiste à sélectionner parmi toutes les partitions possibles la partition optimale en minimisant un critère le schéma ci-dessous montre le processus générale de l'algorithme génétique (figure 3.6).

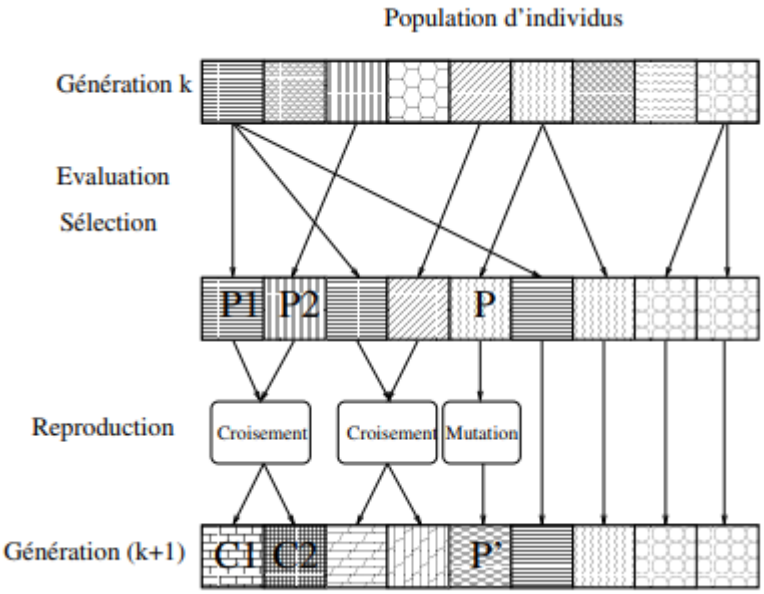


Figure 3.6 Schéma globale de AG [20].

3.3.1 Le codage proposé

Dans un algorithme génetique ; un individu représente une solution au problème. Dans notre probleme " clustering" on va le représenter par un tableau ou un ensemble de centroïdes de clusters.Pour générer notre problème,on presente chaque individu par une séquence de nombres réels représentant les K centres de clusters. Pour un espace à N dimensions, la longueur d'un individu est constitué de N * K gènes, où les premières N positions (gènes) représentent les N dimensions du premier centre de cluster, les N positions suivantes représentent celles du deuxième groupe centre, et ainsi de suite un individu est créé comme en figure 3.7 [20].

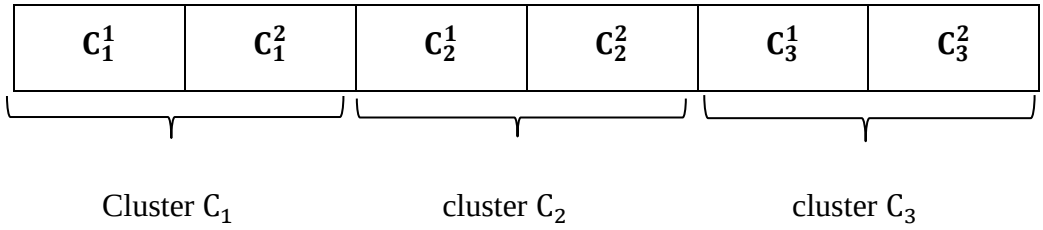


Figure 3.7: Encodage d'un Individu

3.3.2 Fonction objective proposée (Fitness)

Comme on a expliqué dans la méthode BBO dans notre étude on a fixé le nombre de classes depuis le début et on cherche a minimiser la somme des distances entre les points de notre base de données et les centres des classes le WCDS " within-cluster distance sum" pour aborder cet objectif on va utiliser la fonction *ClusteringCost* citée en section (3.2.4) **.**

3.3.3 Initialisation de paramètres d'AG :

Paramètres	Notation	Valeur
Taille de la population	nPop	50
Nombre de générations	Maxit	100
Domaine de définition des variables	[min, max]	[0, 255]
Dimension du problème	nVar	4
Constante de croisement	Pc	0,8
Constante de mutation	Pm	0.3

Tableau 3.4: Valeurs des paramètres de l'algorithme AG

3.3.4 Population initiale

Comme on a vu dans la méthode BBO on va créer une population initiale de façon aléatoire d'un nombre d'individus ; chaque individu est initialisé aléatoirement par k centres de classes alors une population est présentée comme suit:

Pop=
$$\begin{bmatrix} x_{11} & x_{12} & x_{13} & x_{14} & \cdots & x_{1m} \\ x_{21} & x_{22} & x_{23} & x_{24} & \cdots & x_{2m} \\ x_{31} & x_{32} & x_{33} & x_{34} & \cdots & x_{3m} \\ . & . & . & . & & . \\ . & . & . & . & & . \\ . & . & . & . & & . \\ x_{N1} & x_{N2} & x_{N3} & x_{N4} & \cdots & x_{Nm} \end{bmatrix}$$

telque chaque ligne représente un individu

3.3.5. Évaluation

Pour chaque individu on doit calculer son valeur fitness puis trier les individus selon l'ordre croissant de leurs Fitness en utilisant la fonction *Sort* .Des couples de parents P1 et P2 sont sélectionnés en fonction de leurs adaptations ou bien "Fitness".

3.3.6 Sélection

Pour faire une reproduction des nouveaux individus on a besoin de sélectionner quelques individus comme des (Parents); pour faire cette sélection on a choisi la sélection par roulette qui a pour rôle de sélectionner les habitats en fonction de leurs adaptations puis on applique L'opérateur de croisement avec une probabilité Pc [20].

3.3.7 Operateur de croisement

Le croisement a pour but d'enrichir la diversité de la population en manipulant la structure des chromosomes. Classiquement, les croisements sont envisagés avec deux parents et génèrent deux enfants. Pour effectuer ce croisement sur des chromosomes constitués de M gènes, on tire aléatoirement une position dans chacun des parents. On échange ensuite les deux sous-chaînes terminales de chacun des deux chromosomes, ce qui produit deux enfants C_1et C_2 (voir figure 3.7). deux gènes $P_1(i)$ et $P_2(i)$ sont sélectionnés dans chacun des parents à la même position i. Ils définissent deux nouveaux gènes $C_1(i)$ et $C_2(i)$ par combinaison linéaire

$$\begin{cases} C_1(i) = P_1(i) + (1 - \alpha)\, P_2(i) \\ C_2(i) = (1 - \alpha)\, P_1(i) + P_2(i) \end{cases}$$

Où α est un coefficient de pondération aléatoire [20].

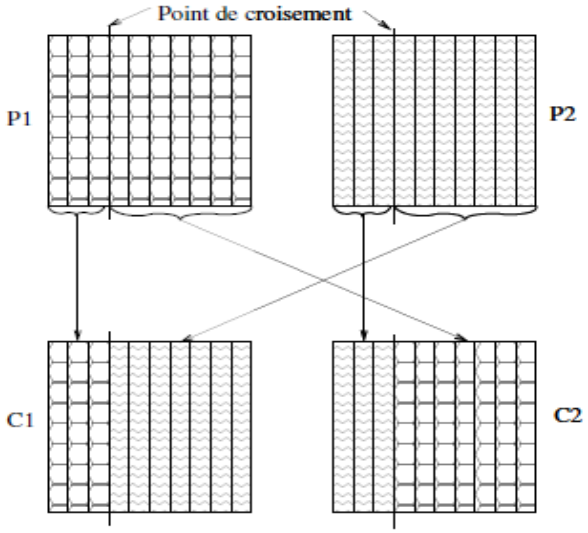


Figure 3.8 : Croisemant a un point

3.3.8 Opérateur de mutation :

L'opérateur de mutation apporte aux algorithmes génétiques la propriété d'ergodicité de parcours d'espace. Cette propriété indique que l'algorithme génétique sera susceptible d'atteindre tous les points de l'espace d'état, sans pour autant les parcourir tous dans le processus de résolution. Pour notre problème, l'opérateur de mutation consiste à tirer aléatoirement un gène dans le chromosome et à le remplacer par une valeur aléatoire (voir figure 3.9), cette valeur doit être choisie dans l'espace de recherche [20].

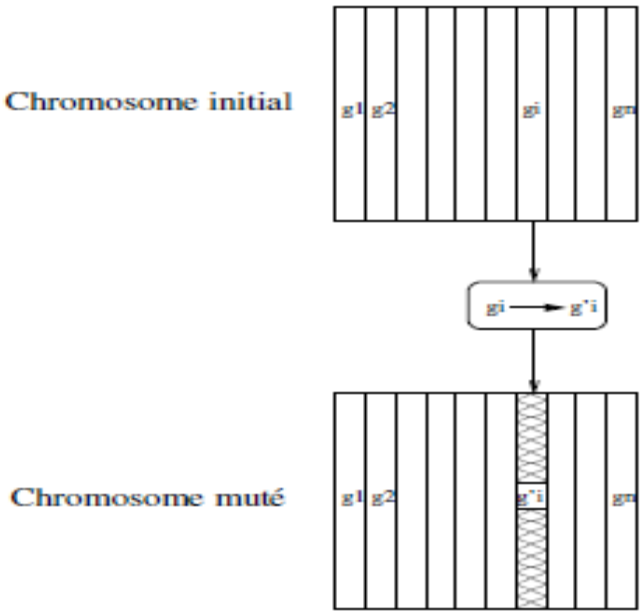


Figure 3.9 : Principe de l'opérateur de mutation

Les enfants C_1, C_2 et les individus muté P_0 sont ensuite évalués avant insertion dans la nouvelle population (la figure 3.6) présente le cas où les enfants et les individus mutés remplacent les parents). Différents critères d'arrêt de l'algorithme peuvent être choisis:

– Le nombre de générations que l'on souhaite exécuter peut être fixé a priori. C'est ce que l'on est tenté de faire lorsque l'on doit trouver une solution dans un temps limité.

– L'algorithme peut être arrêté lorsque la population n'évolue plus ou plus suffisamment rapidement [20].

3.4 Conclusion

Dans ce chapitre, d'abord nous avons présenté en détail les deux métaheuristiques : l'algorithme BBO et l'algorithme génétique GA, ses principes et ses paramètres. Ensuite nous avons adapté chacune d'elle pour réaliser une tache de classification non supervisée.

CHAPITRE 4

REALISATION ET EXPERIMENTATION

4.1.Introduction

La réalisation est la dernière phase dans tout processus de développement d'un système ou d'un logiciel. Dans ce chapitre, on vise de présenter brièvement les outils et les moyens utilisés pour implémenter nos méthodes proposées dans le chapitre précédent. En particulier, la démarche de conception retenue, l'environnement de programmation choisi, et l'ensemble des interfaces générés par notre application.

4.2. Environnement de développement

Le choix du bon environnement de programmation est très important pour le développement des projets. Cela se fait suivant plusieurs facteurs: la puissance de compilation, la facilité d'utilisation, la disponibilité de plusieurs fonctionnalités, la communication avec d'autres environnements, etc. pour implémenter notre système nous avons utilisé les outils suivants:

4.2.1.Système d'exploitation

Pour que notre travail atteigne l'objectif qu'on visait, on a pris l'initiative d'exploiter et d'implémenter notre algorithme sur la version : Windows 7. Ce choix se traduit par l'efficacité de cet environnement en ce qui concerne la structure d'interaction événementielle qu'elle dispose pour communiquer avec des applications actives, ainsi que les ressources de la machines qu'il offre aux différentes applications, enfin, son système d'allocation de mémoire qui est un des meilleurs présents dans ce domaine.

4.2.2 language de programmation MATLAB R2013b

Pour l'implémentation de notre application, nous avons choisi MATLAB « matrix laboratory » est un langage de programmation de quatrième génération émulé par un environnement de développement du même nom ; son environnement de bureau est adapté pour l'analyse par itération et les processus de conception avec un langage de programmation permettant d'exprimer directement les mathématiques sous forme de tableaux et de matrices. il est utilisé à des fins de calcul numérique. MATLAB permet de manipuler des matrices, d'afficher des courbes et des données, de mettre en œuvre des algorithmes, de créer des interfaces utilisateurs, Les toolboxes MATLAB sont développées de façon professionnelle, Les applications MATLAB nous permettent de tester différents algorithmes sur vos données.

Itérez jusqu'à obtenir les résultats attendus, puis générez automatiquement un programme MATLAB pour reproduire ou automatiser notre travail. MATLAB nous permet d'Exécutez nos analyses sur des clusters, des GPU et sur le Cloud avec un minimum de modifications au niveau du code. nous n'avons pas besoin de réécrire notre code ou de vous familiariser avec la programmation Big Data et les techniques hors mémoire.

La Figure 4.1 présente l'interface de MATLAB R2013b

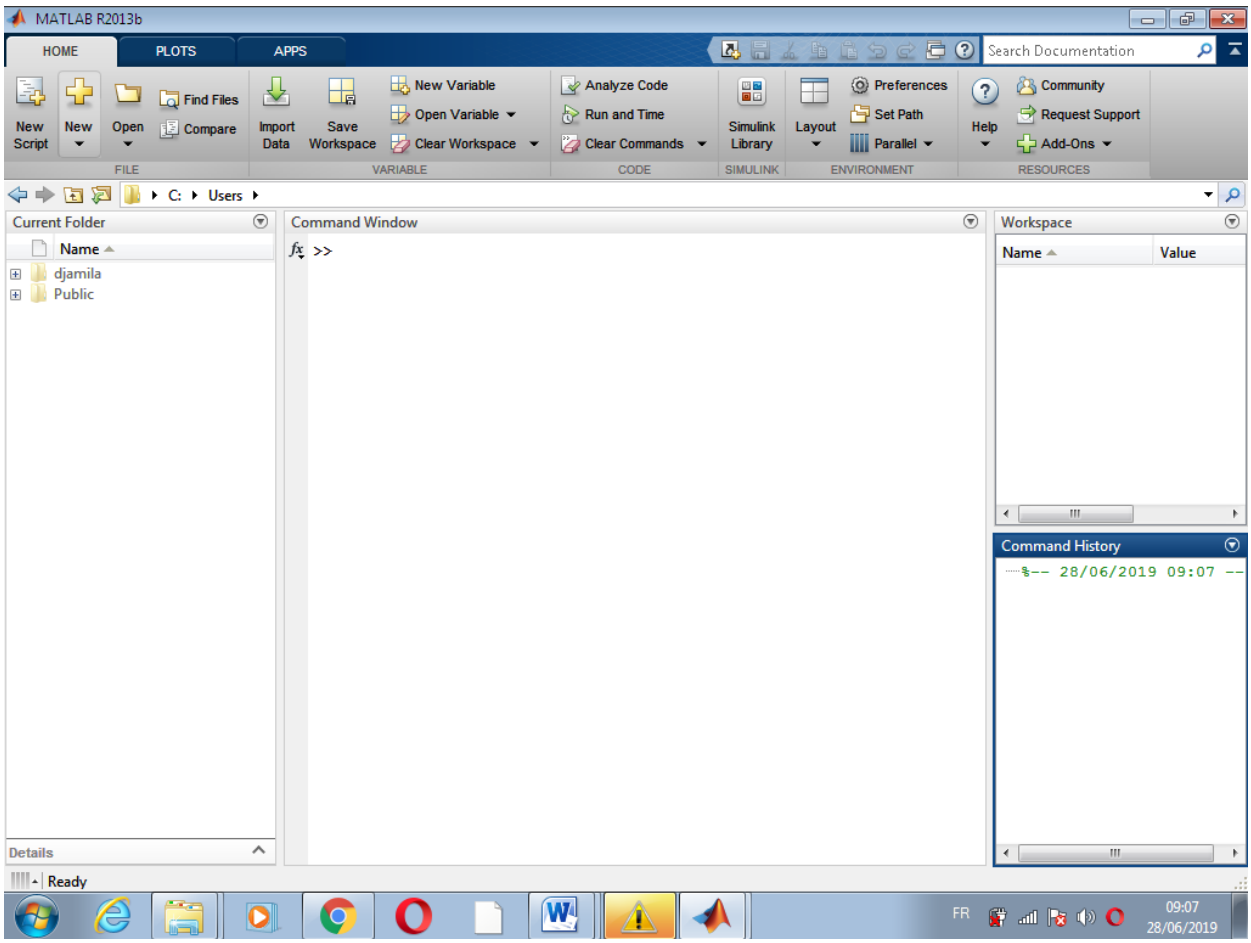


Figure 4.1 : Interface de MATLAB R2013b

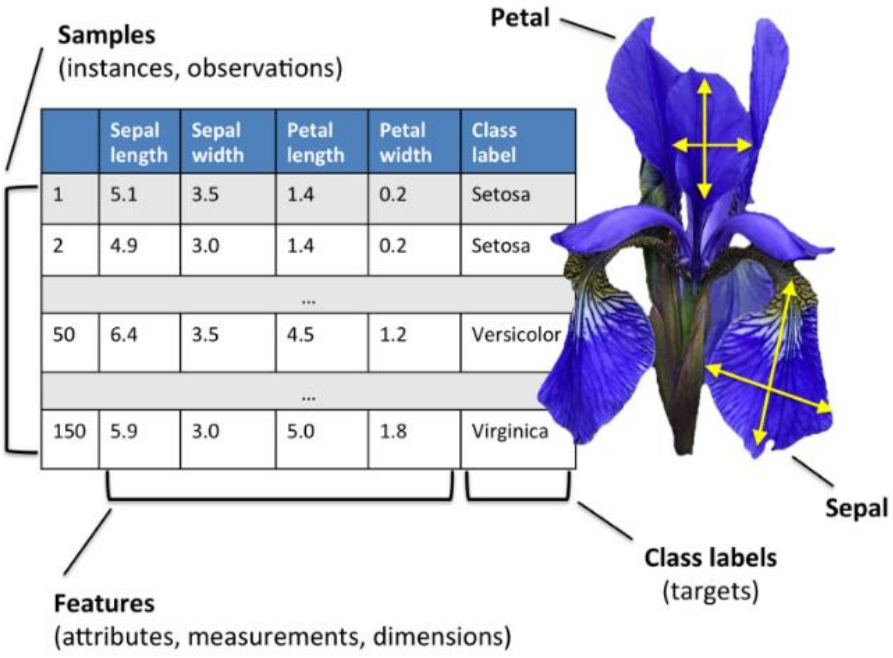
.

4.3Résultats expérimentaux

Nous allons présenter les résultats obtenus par les algorithmes proposés pour prouver leurs importances d'un point de vue de la recherche d'une solution approchée au problème de clustering. Ensuite et afin d'évaluer la performance des algorithmes, une comparative est menée entre nos deux algorithmes sur des instances de différentes jeux de données .

4.3.1Les Données utilisées

Pour évaluer la qualité des metaheuristiques proposées (BBO et AG) on va les tester sur un jeu de données "dataset" de fleurs d' IRIS bien connu. Ce jeu de données comprend un total de 150 observations, réparties de manière égale entre les trois espèces de fleurs d'iris (setosa,virginica et versicolor). Quatre caractéristiques sont mesurées pour chaque observation (c.-à-d. la longueur et la largeur du sépale et du pétale, en centimètres) voir figure 4.1.



	Sepal length	Sepal width	Petal length	Petal width	Class label
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
...					
50	6.4	3.5	4.5	1.2	Versicolor
...					
150	5.9	3.0	5.0	1.8	Virginica

Figure 4.2 :vue sur dataset IRIS

4.3.2 Les résultats obtenus par l'algorithme génétique :

Dés l'execution de la fonction main s'affiche la boite de dialogue figure 4.3

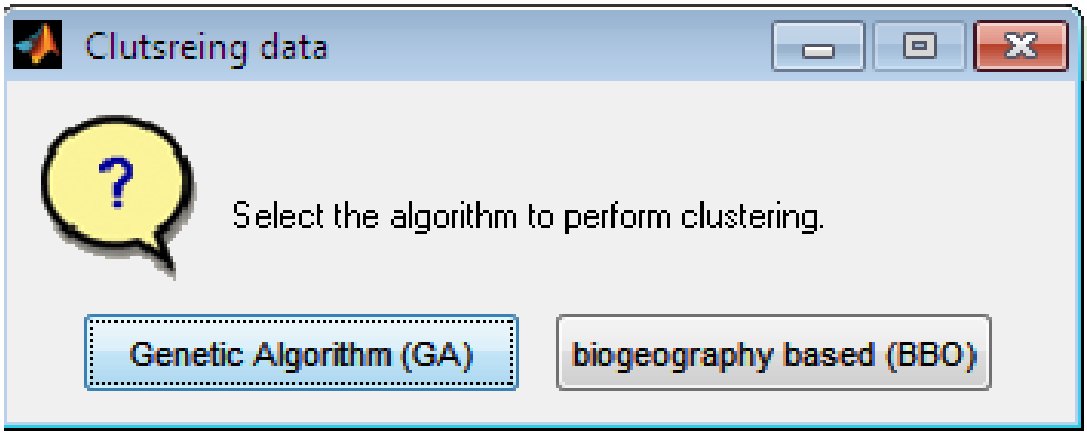


Figure 4.3 Boite de dialogue a deux choix (GA et BBO)

-En appuyant sur le botton Genetic Algorithm(GA) les resultats obtenus par l'algorithme --génétique s'affichent.

- La figure 4.4 illusttre une solution optimale (les troix classes de jeu de donnée IRIS) detectés par (AG) en colours rouge bleu et vert.

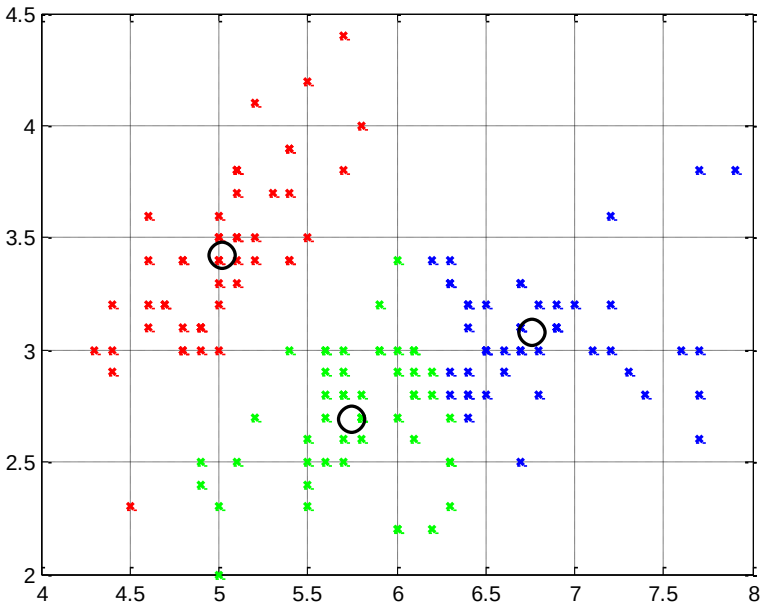


Figure 4.4 : Solution a deux dimensions pour le jeu de données IRIS obtenue par AG

- La figure 4.5 montre la courbe de convergence de l'algorithme genetique vers la valeur optimale on observe la réduction de fonction de cout avec le nombre d'itération ,et que l'algorithme genetique atteint la valeur optimale BestCost 62.2176 aprés l'iteration 72.

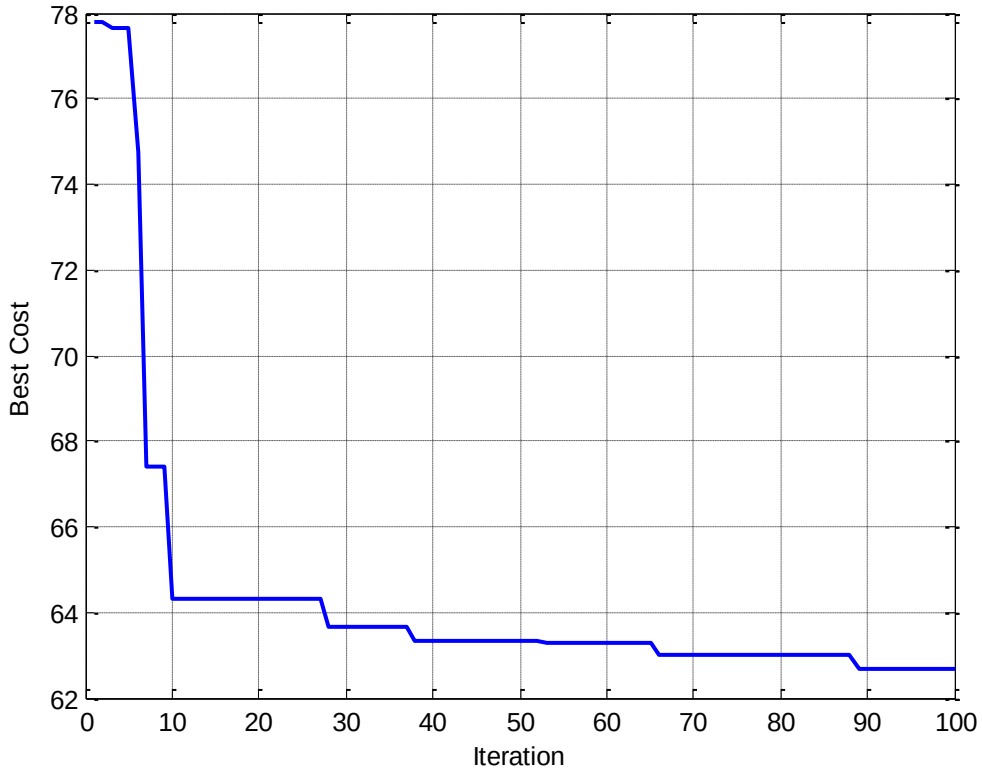
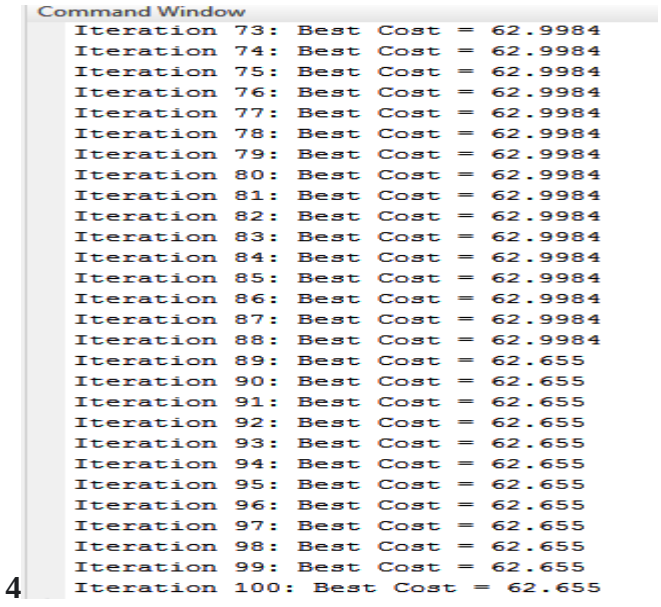


Figure 4.5 : la courbe de convergence vers la valeur optimale obtenue par AG.



4

Figure 4.6 : la réduction de fonction fitness pendant les dérnieres itérations obtenue par AG.

- Discusion de resultats :On remarque que l'algorithme génétique atteint la valeur optimale 62.655 pour ses paramètres après 89 générations,

4.3.3 Les resultats obtenus par l'algorithme Biogeography based optimisation:

En appuyant sur le botton biogeography based optimisation (BBO) dans la boite de dialogue les resultats obtenus par l'algorithme BBO s'affichent les figures 4.7 ,4.8 ,4.9.

La figure 4.7 illusttre une solution (classification),ou bien les troix classes de jeu de donnée IRIS detecté par (BBO) en colour rouge bleu et vert.

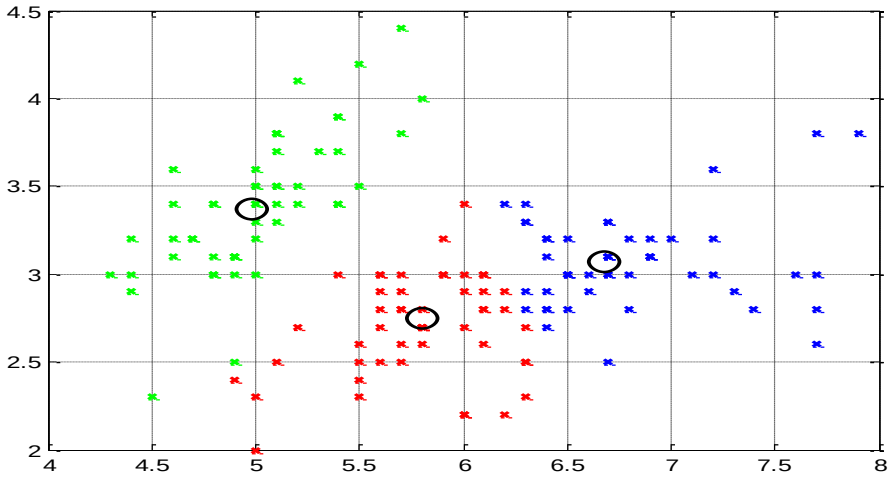


Figure 4.7 : Solution a deux dimensions pour le jeu de données IRIS obtenue ar BBO.

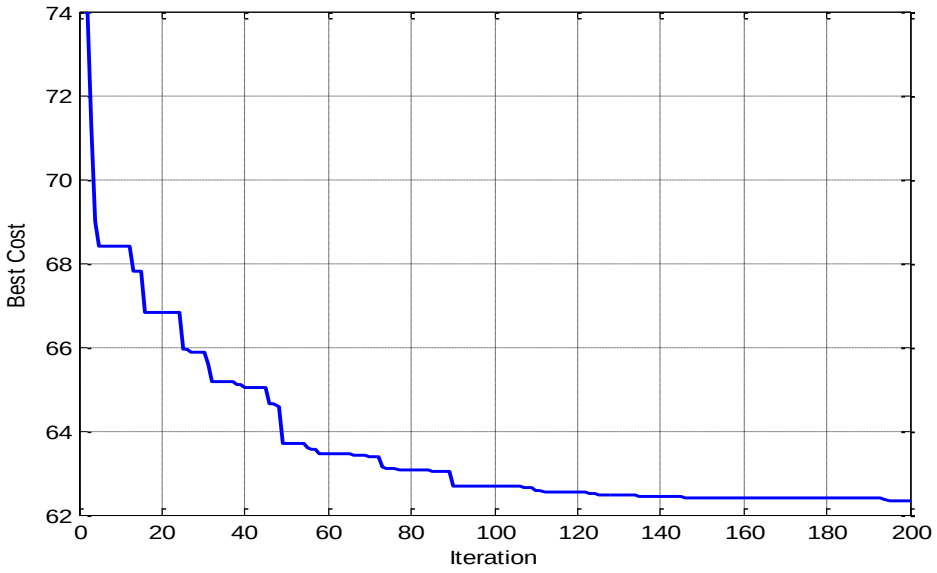


Figure 4.8 : la courbe de convergence vers la valeur optimale obtenue par BBO.

- On observe que la BBO a atteint les valeurs optimales Best Cost = 62.2173 pour ses paramètres après 96 générations.

```
Command Window
   Iteration 73: Best Cost = 62.2179
   Iteration 74: Best Cost = 62.2179
   Iteration 75: Best Cost = 62.2179
   Iteration 76: Best Cost = 62.2179
   Iteration 77: Best Cost = 62.2179
   Iteration 78: Best Cost = 62.2179
   Iteration 79: Best Cost = 62.2179
   Iteration 80: Best Cost = 62.2179
   Iteration 81: Best Cost = 62.2178
   Iteration 82: Best Cost = 62.2178
   Iteration 83: Best Cost = 62.2178
   Iteration 84: Best Cost = 62.2178
   Iteration 85: Best Cost = 62.2178
   Iteration 86: Best Cost = 62.2178
   Iteration 87: Best Cost = 62.2178
   Iteration 88: Best Cost = 62.2178
   Iteration 89: Best Cost = 62.2178
   Iteration 90: Best Cost = 62.2175
   Iteration 91: Best Cost = 62.2175
   Iteration 92: Best Cost = 62.2175
   Iteration 93: Best Cost = 62.2174
   Iteration 94: Best Cost = 62.2174
   Iteration 95: Best Cost = 62.2174
   Iteration 96: Best Cost = 62.2173
   Iteration 97: Best Cost = 62.2173
   Iteration 98: Best Cost = 62.2173
   Iteration 99: Best Cost = 62.2173
   Iteration 100: Best Cost = 62.2173
fx >>
```

Figure 4.9 : la réduction de fonction fitness pendant les dérnieres itérations obtenue par BBO

4.3.4 Étude Comparative

A partir de ces résultats on peut juger que la méthode BBO est meilleure que la méthode GA en matière de la qualité de la solution trouvée mais je propose que les méthodes doivent etre hybridées par d'autres algorithmes pour accélérer la recherche .

Conclusion

Dans ce chapitre nous avons appliqué la methode GA et BBO pour la classification non supervisée de données. Les deux algorithmes sont tester sur des données de la littérature, en particulier IRIS. Les résultats obtenus montrent la capacité des techniques bio-inspirées pour la résolutin de ce type de problemes notament pours des données de tailles importantes.

CONCLUSION GENERALE

Conclusion

Le clustering est l'une des tâches majeures dans différents domaines de recherche. Son but principal est d'identifier et extraire des groupes significatifs dans les données. Pour réaliser cette tâche, plusieurs méthodes sont décrites dans la littérature.

Au cours de notre projet de fin d'étude, nous avons étudié deux métaheuristiques inspirées de la nature : l'algorithme BBO et l'algorithme génétique GA, L'objectif était de comprendre le fonctionnement de chaque méthode, Ensuite nous avons adapté chacune d'elle pour résoudre le problème de clustering pour des données numeriques. Afin de mieu tester leurs efficacité, nous avons choisi le jeu de données IRIS comme objet de test.

Les resultas obtenus montrent l'avantage de l'application des metaheuristiques d'optimisation dans le domaine de fouille de données ; en particulier au niveau de la tache de classification dans le cas où la taille du probleme est considérable.

Comme perspective à ce travail, il est recommender d'appliquer d'autres techniques bio-inspirées pour la classification de données. Egalement, il est possible de concevoir des methodes hybrides basées sur les metaheuristiques pour ameliorer la qualité de la classification et de réduire le temps d'exécution.

Bibliography

[01] A.K.Jain, M. N. Murty, and P. J. Flynn. Data ordonnancement: a review. ACM Computing Surveys, 31(3):264_323, 1999.

[02] Cherif Sadfi, 2002, Problèmes de classification avec Minimisation des Encours. Thèse Ph.D, Institut National Polytechnique de Grenoble.

[03] Christophe Rapine et denis Trystram, 2002, Théorie de la Complexité, Notes de cours, ENSGI – INP Grenoble.

[04] D.Simon. Biogeography-based optimization. IEEE Transactions on Evolutionary Computation, 12(6): 702–713, 2008.

[05] E. Forgy. Cluster analysis of multivariate data : efficiency versus interpretability of classifications. Biométrics, 21 :768_780, 1965.

[06] F. Glover and M. Laguna. Tabu Search. Kluwer Academic Publishers.

[07] Francois-Gérard, Souquet Amédée, Radet algorithmes genetiques, 2004.

[08] Guillaume Cleuziou, Une méthode de classification non-supervisée pour l'apprentissage de règles et la recherche d'information, décembre 2004, Université d'Orléans.

[09] Gonzalez T. (1985). Clustering to minimize the maximum intercluster distance. Theoretical Computer Science.

[10] Guénaël CABANES, Classification non supervisée à deux niveaux guidée par le voisinage et la densité, 03/12/10, Université Paris

[11] Hodson (F. R.), Sneath (P. H. A.) et Doran (J. E.). { Some experiments in the numerical analysis of archaeological data. Biometrika, vol.53, 1966, pp. 311{324.

[12] Holland, J. H. "Adaptation in Natural and Artificial Systems". University of Michigan Press, Ann Arbor, 1975. cité dans [SIM 13a]. algorithm. Applied Statistics, 28(1) :100_108, 1979. continue, universite paris-est créteil, 2013.

[13] Ilhem Boussaid, perfectionnement de métaheuristique pour l'optimisation [01] Gérard Ramstein, Application de techniques de fouille de données en Bio-informatique, l'Ecole polytechnique de l'université de Nantes, 2012.

[14] Inderjit S. Dhillon, Subramanyam Mallela, and Rahul Kumar. Enhanced word clustering for hierarchical text classi_cation. In KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, pages 191_200, New York, NY, USA, 2002. ACM.

[15] J. A. Hartigan and M. A. Wong. Algorithm AS 136 : A k-means clustering

[16] J.Dréo, A.Péreowski, P.Siarry et E.Taillard, Métaheuristiques pour l'optimisation difficile. Editions Eyrolles. 40, 2003.

[17] Jalam, Radwan . (2003) "Apprentissage automatique et catégorisation de textes multilingues". Thèse de doctorat, Université Lumière Lyon 2 URL : http://www.agrocampus-rennes.fr/tice/jalam/these/these_radwan.ps.gz

[18] Kettenring (J. R.), Rogers (W. H.), Smith (M. E.) et Warner (J. L.). { Cluster analysis applied to the validation of course objectives. J. Educ. Statist., vol.1, 1976, pp. 39,57}

[19] Losee, R.M. (1998). Text retrieval and filtering analytic models of performance. Kluwer Academic Publishers.

[20] Nicolas Durand , Algorithmes Génétiques et autres méthodes d'optimisation appliqués à la gestion de trafic aérien ,2004 ,Thèse , l'institut national polytechnique de Toulouse

[21] Nicoleta ROGOVSCHI, Classification à base de modèles de mélanges topologiques des données catégorielles et continues, Université Paris 13 - Institut Galilée

[22] Pascal Rebreynd, 1999, Algorithmes Génétiques Hybrides en Optimisation Combinatoire, Thèse Ph.D, Ecole Normale Supérieure de Lyon.

[23] Rabiner (L. R.), Levinson (S. E.), Rosenberg (A. E.) et Wilpon (J. G.). { Speaker independent recognition of isolated words using clustering techniques. IEEE Trans. Accoust. Speech Signal Process., vol.27, 1979, pp. 336_349}.

[24] R.MacArthur & E. Wilson, The Theory of Biogeography, Princeton University Press, Princeton, NJ, 1967

[25] R.Ricklefs & G. Miller. Ecologie. Éd. de Boeck, 2005. ISBN 2-7445-0145-X.

[26] Siegel (J. H.), Goldwyn (R. M.) et Friedman (H.P.). { Pattern and process of the evolution of human septic shock. Surgery, vol.70, 1971, pp. 232{245.

[27] Spath H. Clustering algorithms. John Wiley and Sons, New York, NY, 1975.

[28] S.B. Kotsiantis, P. E. Pintelas, "Recent Advances in Clustering: A Brief Survey", WSEAS Transactions on Information Science and Applications,Vol 1(1), pp.73–81,2004.

[29] Siarry &Collette,2004.

[30] Sylvain Grassin & Pauline Regnault, Classification par une méthode de Clustering basée sur la densité, Département d'Enseignement et de Recherche en Informatique & Technologies Urbaines, Ecole d'ingénieurs de la ville de Paris.

[31] [Taillard, 2002] É. D. Taillard. Principes d'implémentation des métaheuristiques". In Jacques

[32]. Teghem & Marc Pirlot, editors, Optimisation approchée en recherche opérationnelle,. Hermès, 2002.

[33] Vincent Barichard, 2003, Approches Hybrides pour les Problèmes Multi objectifs, Thèse Doctorat, École Doctorale d'Angers.

ملخص

يتطلب تحليل واستغلال كميات كبيرة من البيانات تقنيات متخصصة ، مثل التنقيب في البيانات و التي تستخرج المعرفة من البيانات لإنشاء نموذج مفهوم و مثير للاهتمام و من بين المهام الوصفية لعملية التنقيب في البيانات هو التصنيف غير خاضع للرقابة والذي يعمل على تجميع البيانات المتشابهة في نفس الفئات دون أي معرفة مسبقة بعدد الفئات أو علاماتها ، وذلك بناءً على فكرة المسافة بين البيانات لتحقيق أقصى قدر من التشابه داخل الفئة الواحدة وتقليل التشابه بين الفئات المختلفة ولهذا سنستعمل خوارزميات البحث المعمق المستوحاة من الطبيعة وتطور الكائن البشري من أجل تطبيق مشكلة التصنيف على قاعدة بيانات كبيرة .الهدف من هذا العمل هو تنفيذ خوارزمية بحث معمق جديدة تعتمد على نظرية الجغرافيا الحيوية لدراسة التوزيع المكاني للأنواع الحية (النباتية والحيوانية) والخوارزمية الوراثية لإيجاد تصنيف مثالي لقاعدة البيانات المعروفة بزهرة الأيرس.

الكلمات المفتاحية: التنقيب في البيانات – التصنيف الغير مراقب - تحليل البيانات - مجموعة بيانات زهرة الأيرس – خوارزميات البحث المعمق - التحسين الحيوي استنادًا إلى الجغرافيا الحيوية- تعليم الألة

Abstract

The analysis and exploitation of large amounts of data requires specialized techniques, such as datamining, which extracts knowledge from data to create an interesting and understandable model. One of the descriptive tasks of the latter is the clutering (unsupervised classification) of grouping the most similar data in the same classes without any prior knowledge of the number of classes or their labels, based on the notion of distance between data to maximize intra class similarity and minimize interclass similarity.

To do this, we will use metaheuristics inspired by the nature and evolution of the human being in order to transform the clustering problem to a large data space. The objective of this work is to use a new metaheuristic based on the theory of biogeography consists of the study of the spatial distribution of living species (plant and animal) and a genetic algorithm to find an optimal classification to our game of data" flower Iris".

Key words: Datamining - Clustering - Unsupervised classification - Data analysis - Dataset Iris clustering- Metaheuristics -Biogeography based optimization_Machine learning.

Résumé

L’analyse et l’exploitation de grande quantité de données nécessite des techniques spécialisées, tel que le datamining qui permet d’extraire des connaissances à partir de données afin de créer un modèle intéressant et compréhensible. L’une des taches descriptives de ce dernier est le clutering (la classification non supervisée) qui consiste à regrouper les données les plus similaires dans les mêmes classes sans aucune connaissance apriori sur le nombre de classes ni leurs étiquettes, en basant sur la notion de distance entre les données pour

maximiser la similarité intra classes et minimiser la similarité inter classes.Pour ce faire, nous allons utiliser les metaheuristiques inspirées de la nature et de l'evolution de l'etre humain afin de transformer le problème de clustering à un grand espace de données.L'objectif de cette travail est d'utiliser une nouvelle métaheuristique basée sur la théorie de la biogéographie consiste en l'étude de la répartition spatiale des espèces vivantes (végétales et animales) et une algorithme génétique pour trouver une classification optimale a notre jeux de données"la fleur de Iris".

Mots clés: Fouille de données – Classification non supervisée –Analyse des données – base de données Iris- Métaheuristique – Optimisation basé sur Biogeographie _ Apprentissage automatique.