République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Abbés Laghrour Khenchela
Faculté des Sciences et de la Technologie
Département de Mathématiques et Informatique

Mémoire pour obtenir le diplôme de

Master en Informatique

Spécialité : Génie logiciel et systèmes distribués.

Une approche d'optimisation par essaim de particules quantique pour la sélection d'attributs dans la classification de données

Présenté Par :

Sihem BEBKHALED

Jury :

Dr Rafik MAHDAOUI	(MCB)	Univ.Abbés Laghrour Khenchela	Président
Dr Hichem HOUASSI	(MCB)	Univ.Abbés Laghrour Khenchela	Rapporteur
Mr Makhlouf LEDMI	(MAB)	Univ.Abbés Laghrour Khenchela	Examinateur

Soutenu le 01 juillet 2017

Dédicace

"... Sometimes it is the people no one imagines anything of who
do the things that no-one imagine."

Alan Turing

Remerciements

Avant toute chose, je remercie Dieu, le tout puissant, de m'avoir donné la force et la patience pour mener ce travail à bout.

J'exprime mes profonds remerciements et mes vives reconnaissances à Docteur Hichem HOUASSI Maître de conférence à l'Université Abbès Laghrour de Khenchela, d'avoir dirigé ce travail avec une grande rigueur scientifique tout en préservant mon esprit d'intuitive, ainsi que ses conseils et pour la confiance qu'il m'a accordé afin de réaliser ce travail.

J'aimerais remercier les membres de jury de m'avoir fait l'honneur de participer à mon jury.

Bien entendu, je remercie mes parents et toute la famille pour le soutien moral qu'ils m'ont apporté tout au long de ce travail.

Résumé

La sélection d'attributs est une étape de prétraitement qui joue un rôle important dans la fouille de données, elle permet de réduire le nombre d'attributs (features) des bases de données qui contient des ensembles volumineux d'attributs, en éliminant les attributs redondants et non pertinentes pour faciliter les tâches de fouille de données et surtout la tâche de classification supervisée, toutes en essayant de maintenir ou d'améliorer la performance de classifieur.
La recherche d'un sous-ensemble d'attributs est un problème d'optimisation NP-difficile qui doit être résolu par les méthodes approchées (métaheuristiques).
Nous nous sommes intéressés aux métaheuristiques issues de l'intelligence en essaims de particules PSO (Particle swarm optimisation) pour la sélection d'attributs dans la classification supervisée de données.
Dans ce travail, nous avons proposé un algorithme de sélection de sous-ensemble d'attributs pertinents dénoté BC-QPSO et issue d'une hybridation entre une métaheuristique d'optimisation par essaim de particules (PSO) binaire quantique et l'algorithme de sélection clonale isuue du système immunitaire artificiel, les expérimentations ont été effectuées en utilisant des bases de données publié dans le site de l'UCI (University of California, Irvine), et les résultats expérimentaux de notre approche sont comparés à ceux obtenus par deux algorithmes BPSO et BQPSO proposées dans la littérature, ces résultats obtenus sont compétitifs et montre l'efficacité de notre approche proposée par rapport aux autres.

Mots clés : Sélection d'attributs, Métaheuristiques, Optimisation par essaim de particules quantique (BQPSO), Sélection clonale.

Abstract

Feature selection is a pre-processing step that plays an important role in data mining, allowing it to search a reduced subset of features from a large set of features by eliminating redundant and irrelevant features for performing the supervised classification task, all trying to maintain or improve classifier performance.
The search for a subset of features is an NP-difficult optimization problem that can be solved by metaheuristics, we have been interested in the metaheuristics resulting from the intelligence in swarms for the features selection.
In this work, we proposed a Binary Clonal Quantum Particle Swarm Optimization algorithm denoted as BC-QPSO for selecting a subset of relevant features resulting from a hybridisation between a metaheuristic "Binary Quantum Particle Swarm Optimization" and "Artificial immune system" Using its clonal selection algorithm.
The experiments were carried out using UCI databases (University of California, Irvine), and the experimental results of our approach are compared with those obtained by BPSO and BQPSO algorithms proposed in the literature, the results obtained are competitive and show the efficiency of our BC-QPSO algorithm.

Keywords : Features Selection, Metaheuristics, Binary Quantum Particle Swarm Optimization (BQPSO), Clonal Selection algorithm.

Table des matières

Liste des tableaux

Liste des figures

Liste des algorithmes

Liste des abréviations

$BC-QPSO$: Binary Clonal Quantum-Behaved Particle Swarm Optimization

$BPSO$: Binary Particle Swarm Optimization

$BQPSO$: Binary Quantum-Behaved Particle Swarm Optimization

Gbest : Meilleure position connue par toutes les particules de l'essaim

MTC : Meilleur Taux de Classification

NMA : Nombre Moyen d'Attributs Sélectionnées

NTA : Nombre Total d'Attribut

Pbest : Meilleure position connue par un particule

PSO : Particle Swarm Optimization

$TCTA$: Taux de Classification utilisant Tout les Attributs de la base de test

TMC : Taux Moyenne de Classification

Introduction générale

La fouille de données est un processus dont le but est l'extraction de connaissances, la découverte de nouvelle corrélations et modèles en analysant une grande quantité de données, il existe différentes techniques de fouille de données, y compris la classification supervisée, le clustering, la recherche des règles d'association.
La technique la plus importante est la classification supervisée où l'objectif est de construire un ou plusieurs modèles sur des données d'apprentissage, qui peut prédire correctement la classe d'un nouveau objet, la complexité de cette tâche s'est accrue suite à l'augmentation du taille de l'ensemble d'apprentissage disponibles, pour cela il est fondamental de mettre en place des techniques de traitement de données qui permettent une meilleure compréhension de la qualité des connaissances disponibles dans ces données.
La sélection de caractéristiques ou bien d'attributs est l'une des solutions les plus classiques permettant d'apporter des éléments de réponse à ce problème, son objectif est de sélectionner un sous-ensemble optimal d'attributs pertinents et réduit selon un critère prédéfini, les premières approches de sélection d'attributs s'appuient sur des méthodes classiques.
Actuellement, le problème de sélection d'attributs devient un problème d'optimisation Nb-difficile, malgré l'existence des méthodes classiques pour traiter ce problème, ces méthodes restent limitées et moins efficaces, la tendance de la recherche dans le domaine de la sélection d'attributs est orientée vers les métaheuristiques en particulier celles issues de l'intelligence en essaim, ces méthodes ont suscité un grand intérêt dans les deux dernières décennies.
Les métaheuristiques constituent une classe de méthodes qui fournissent les solutions de bonnes qualités en un temps raisonnable à des problèmes difficiles pour lesquels il n'existe pas des méthodes classiques plus efficaces, elles sont généralement des méthodes stochastiques et itératives qui progressent vers un optimum global en évaluant une fonction objective.
Suite à une étude des travaux relatifs aux méthodes métaheuristiques hybrides, nous avons constatées le manque de propositions d'hybridations de méthodes de «l'optimisation par essaim de particules quantique» et l'optimisation par « le système immunitaire artificiel », Donc il faut penser à proposer des approches hybrides efficaces.
Dans le cadre de ce travail, nous nous intéressons au problème de la sélection d'attributs pour la classification, et plus précisément à l'utilisation d'approches d'intelligence en essaim pour effectuer la sélection d'attributs dans un contexte de classification.
En effet, notre thème de ce mémoire se base sur la proposition d'une nouvelle approche hybride, qui consiste à combiner la version quantique de l'algorithme d'optimisation par essaim de particules et l'algorithme de sélection clonale, nous appelons cette approche « Binary Clonal Quantum Particle Swarm Optimisation » (BC-QPSO).
Cette approche vise d'une part à réduire la taille de l'ensemble d'attributs utilisés avec une convergence rapide vers la solution optimale globale et d'autre part à améliorer la tâche de classification (augmenter le taux de classification).

Démarche de travail :

Après cette section introductive, ce mémoire et organisée en 3 chapitres :

Chapitre1 : Dans ce chapitre on va présenter la fouille de données comme étant une tâche essentielle dans le processus de l'extraction de connaissances à partir des données (ECD), ses motivations, ainsi

que ses tâches en se concentrant sur la classification supervisée.

Chapitre2 : Nous présenterons un état de l'art sur le problème d'optimisation étudié dans ce travail qui est « la sélection d'attributs » et le processus général de la sélection, ainsi que les différentes méthodes proposées dans la littérature pour résoudre ce problème, avec une classification de ces dernières, en se concentrant sur la classe des métaheuristiques connues dans la littérature avec une présentation des principales métaheuristiques surtout les variétés des algorithmes d'optimisation par essaim de particules.

Chapitre3 : Ce chapitre est consacré à la résolution du problème de sélection d'attributs par une hybridation entre la version binaire quantique de l'algorithme d'optimisation par essaim de particules (BQPOS) et l'algorithme de sélection clonale avec une implémentation de l'approche proposée et on la comparer avec deux variantes de PSO l'approche BPSO et BQPSO, notre travail sert à faire une comparaison de performances de notre approche proposées par rapport aux celles de la littérature, en présentant les résultats expérimentaux obtenus.

Et enfin en conclure.

Chapitre 1

Fouille de données et classification

1.1 Introduction

La fouille de données est l'une des technologies émergentes qui changeront le monde au XXIe siècle, cette technologie apparait en 1990 aux États - Unis comme un domaine de recherche en plein essor visant à exploiter les grands quantités de données collectées chaque jours[65].

Ce domaine pluridisciplinaire se situe au confluent de l'intelligence artificielle (notamment de l'apprentissage automatique, des statistiques et des bases de données)[38].

La classification est la tâche la plus importante de la fouille de données, consiste à examiner des caractéristiques d'un objet afin de l'affecter à une classe d'un ensemble donné[44].

1.2 Fouille de données

La fouille de données est le cœur du processus d'extraction de connaissance qui consiste à rechercher et à extraire de l'information utile et inconnue a partir de données stockées dans des bases ou des entrepôts de données, dans le but d'y découvrir de l'information implicite[65].

« Le datamining est défini comme un processus non trivial consistant à identifier dans les données, des schémas nouveaux, valides, potentiellement utiles et surtout compréhensibles et utilisables »[69].

« Le datamining consiste à l'exploration et l'analyse, par des moyens automatiques ou semi automatiques, d'un large volume de données afin de découvrir des tendances ou des règles »[17].

Il est ici important de différencier les trois termes suivants[69] :

- **Données :** Une donnée est le résultat direct d'une mesure.
- **Information :** Est une donnée à laquelle un sens et une interprétation ont été donnés.
- **Connaissance :** Est le résultat d'une réflexion sur les informations analysées.

1.2.1 Motivations de la fouille de données

Le développement récent de la fouille de données (Datamining) lie a plusieurs facteurs[51] [18] :

- **Explosion des données**

  Les outils de collecte automatique des données et les bases de données conduisent à d'énormes masses de données stockées dans des entrepôts.

La prise de conscience de l'intérêt commercial pour l'optimisation des processus de fabrication, vente, gestion, logistique, etc.

Une puissance de calcul importante est disponible sur les ordinateurs de bureau ou même à domicile. L'accès aux réseaux de taille mondial, ces réseaux ayant un débit sans cesse croissant, qui rendent le calcul distribue et la distribution d'information sur un réseau d'échelle mondiale viable.

- **Amélioration de productivité**

Gestion des stocks, des ventes d'un groupe afin de prévoir et anticiper au mieux les tendances du marché. Suivi des fichiers clients d'une banque, d'une assurance, associes a des données socio-économiques.

Web mining et comportement des internautes : application des techniques de datamining au contenu, a la structure, aux usages (ressources du web).

Image mining (analyse et traitement d'images satellites : données spatiales, temporelles, spectrales ... etc.)

1.2.2 Domaines d'application

Actuellement, la fouille de donnée (Datamining) a une grande importance économique. Effectivement, les premières applications se sont faites dans le domaine de la gestion de la relation client qui avait pour but d'analyser le comportement de la clientèle pour la délisser au mieux et lui proposer des produits adaptes[51].
Cependant, cela c'est rependu très rapidement et maintenant utilisée dans différents domaines tels que[51] [18] :

Bancaire : Les institutions financières peuvent obtenir de l'aide de l'extraction de données dans le crédit et le prêt d'informations, un émetteur de carte de crédit peut détecter les fraudes transaction par carte de crédit et aussi dans l'organisme de crédits dans le but de décider l'accorde ou non d'un crédit en fonction du profil du demandeur.

Commercial : Avec optimisation du nombre de places dans les avions, hôtels ... etc.

Médicale : L'analyse du génome, la bioinformatique et dans beaucoup d'autres domaines médicaux.

Marketing et de commerce : Les marketers peuvent faire de la politique pour répondre aux besoins des clients et de comprendre leurs comportements d'achat avec l'aide de l'exploration de données.

Chercheurs : En utilisant des techniques de datamining les chercheurs extraient les connaissances utiles en analysant leurs données et procèdent leurs travaux de recherche.

Éducation : Datamining est très utile dans les établissements d'enseignement, car il y a une grande quantité de données collectées et inutilisés ces données peut être utilise de manière appropriée en utilisant l'exploration de données.

1.2.3 Méthodes de la fouille de données

Les méthodes de fouille de données peuvent être classées selon le type d'apprentissage utilisé [16] :

Fouille supervisé

Comprenant à la fois des données d'entrée et de sortie dont le but est de classer correctement un nouvel exemple, elle s'intéresse à une ou plusieurs variables définies comme étant les cibles de l'analyse[35].

Fouille non supervisé

C'est un processus dans lequel l'apprenant reçoit des exemples d'apprentissage (pas notion de classe) ne comprenant que des données d'entrées, elle n'est pas guidée par un attribut cible spécifique, dont le but est de regrouper les exemples en segment (cluster) similaires ou homogènes [16].

1.2.4 Données manipulées par la fouille de données

Les données manipulées par les techniques de la fouille de données représentent une collection d'objets et leurs attributs ou caractéristiques, ces données possèdent un type qu'il est important de préciser.

En effet, la plupart des méthodes sont sensibles aux données manipulées, par exemple certaines méthodes sont mises en défaut par les données continues alors que d'autres peuvent être sensibles à la présence de données discrètes [16].

Nature de données

- **Données quantitatives :** Valeurs numériques et sommables, discrètes ou continues.
  Exemples : 18, 7654.43, -0.762, 0, 9999.

- **Données qualitatives :** Sa valeur est d'un type défini en extension.
  Exemples : une couleur, une marque de voiture, . . . etc.

Nature de valeurs de données

- **Nominales :** Les valeurs sont des symboles (des noms), aucune relation (ordre ou distance) entre les nominaux n'existe.
  Exemples : féminin, masculin, célibataire, marié, divorcé, veuf.

- **Ordinales :** Une notion d'ordre s'impose sur les ordinaux, mais il n'est pas possible de calculer directement des distances entre des valeurs ordinales, les opérations d'addition et de soustraction ne sont pas possibles.
  Exemples : petit, moyen, grand, très grand.

- **Intervalles :** Les intervalles impliquent une notion d'ordre, et les valeurs sont mesurées dans des unités spécifiques et fixées, la somme, la différence et le produit de deux intervalles ne sont pas possibles.
  Exemples :
  - Une température exprimée en degrés Celsius ou Fahrenheit : T <=23°
  - Un attribut année : (1993, 2015, 2017).

- **Rapport :**Toutes les opérations mathématiques sont autorisées sur les attributs de ce type.
  Exemple :
  - Un attribut distance, on peut comparer deux distances.

1.2.5 Processus d'extraction de connaissances

Les concepts de fouille de données et l'extraction de connaissances (ECD) a partir de données sont parfois confondus et considérés comme synonymes, mais on considère formellement la fouille de données comme une étape essentiel dans le processus d'ECD [65] [69].

D'après [69] un processus d'extraction de connaissance est constitue de quatre phases : Collecte des données, Prétraitement des données, Fouille de données et enfin post-traitement (évaluation et présentation des connaissances).

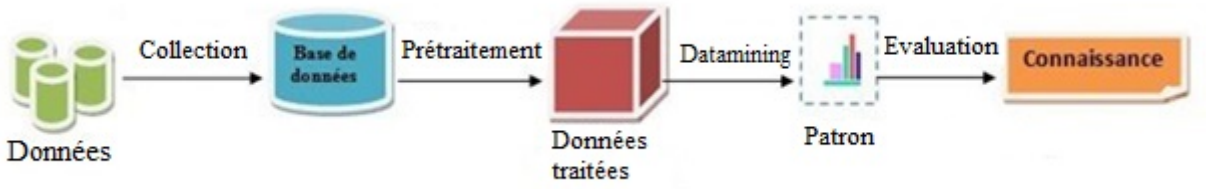

FIGURE 1.1 – Processus général d'extraction de connaissances.

- **La collecte de données :**
  Collecter des informations à partir de plusieurs ressources et les organiser [69].

- **Le prétraitement de données :**
  Un prétraitement est effectué à la fois sur les individus et sur les variables, consiste à nettoyer les données, les mettre en forme, traiter les données manquantes, échantillonner les individus, sélectionner et construire des variables, on obtient ainsi un ensemble de données cibles. Cette phase a une place importante au sein du processus d'ECD car celle qui va déterminer la qualité des modèles construits lors de la phase de fouille de données.
  Elle peut prendre jusqu'à 60 % du temps dédié au processus d'ECD [69].

- **Fouille de données (Datamining) :**
  Cette phase intègre le choix de la méthode d'apprentissage qui va être employée et son paramétrage. Ces choix doivent tenir compte des contraintes liées au domaine étudié ainsi que des connaissances que les experts du domaine peuvent nous fournir.
  L'algorithme sélectionné est alors appliqué aux données cibles dans le but de rechercher les structures sous-jacentes des données et de créer des modèles explicatifs ou prédictifs.
  Certes la fouille de données n'est qu'une étape du processus de l'ECD, mais elle est sans conteste le cœur et le moteur de tous ce processus[65] [69].

- **Post-traitement (Évaluation et présentation) :**
  Cette phase est constitue de l'évaluation qui mesure l'intérêt des modèles extraits, pour aboutir de nouvelles connaissances, et de les présenter comme résultats a l'utilisateur.
  Le processus (ECD) présent est itératif et plusieurs retours en arrière dans les différentes phases sont nécessaires pour affiner les résultats.

La finalité de l'ECD est de pouvoir traiter des données brutes et volumineuses, et à partir de ces données, d'établir des connaissances directement utilisables par un expert ou un non expert du domaine étudié[69].

1.2.6 Tâches de la fouille de données

Le choix des techniques de fouille de données à appliquer dépend de la tâche particulière à accomplir et des données disponibles pour l'analyse [44].
Les principales tâches de fouille de données sont :

La classification (catégorisation) : La tâche supervisée qui consiste à examiner des caractéristiques d'un objet afin de l'affecter à une classe d'un ensemble prédéfini [44] (Voir section 1.3).

La segmentation (clustering) : L'apprentissage non supervisée (on ne définit pas ce qui est entrées et ce qui est sorties), il consiste a former des groupes (clusters) homogènes a l'intérieur d'une population (l'ensemble des enregistrements) par le partitionnement logique de la base de données en clusters, pour cette tâche, il n'y a pas de classes a expliquer ou de valeur a prédire définie a priori, il appartient ensuite a un expert du domaine de déterminer l'intérêt et la signification des groupes ainsi constitues, cette tâche est souvent effectuée avant les précédentes pour construire des groupes sur lesquels on applique des tâches de classification ou d'estimation [27].

L'estimation : Consiste à estimer la valeur d'un champ a partir des caractéristiques d'un objet, le champ à estimer est un champ à valeurs continues. L'estimation peut être utilisée dans un but de classification, il suffit d'attribuer une classe particulière pour un intervalle de valeurs du champ estimé[44].

La prédiction : Consiste à estimer une valeur future, en général les valeurs connues sont historiées, on cherche à prédire la valeur future d'un champ, cette tâche est proche de classification et estimation, les méthodes de classification et d'estimation peuvent être utilisées en prédiction[44].

L'association : La recherche d'association (recherche de règles d'association) est la tâche la plus intéressante du datamining, connue sous le nom d'analyse des associations, c'est également celle qui est la plus répandue dans le monde des affaires, notamment en marketing pour l'analyse du panier de consommation, cette tâche cherche a découvrir les règles de quantification ou de relation entre deux ou plusieurs attributs utiles pour la prise de décision.
Les relations découvertes peuvent êtres représentées sous forme de règles d'association représentées de la forme :

A -> B, c.-à-d. : Si Condition « A » Alors Résultat « B » [61].

1.3 La Classification supervisée

La classification est la tâche la plus connue de la fouille de données et qui semble être une obligation humaine, afin de comprendre notre vie quotidienne, nous sommes constamment classifiés, catégorisés et évalués.

Cette tâche supervisée consiste à étudier les caractéristiques d'un nouvel objet pour lui attribuer une classe prédéfinie, elle est caractérisée par une définition de classes bien précise et un ensemble d'exemples classés auparavant.

L'objectif est de créer un modèle qui peut être appliqué aux données non classifiées dans le but de les classifiées [44].

1.3.1 Définition formelle

Dans la classification supervisée, les classes sont connues et l'on dispose d'exemples (individus) de chaque classe.
Un exemple est un couple (X, Y), où x ∈ X est la description ou la représentation de l'objet et y ∈ Y représente la supervision de x, dans un problème de classification, y s'appelle la classe de x.
Soit un ensemble d'exemples de n données étiquetées :
S = $\{(x_1, y_1),\ldots,(x_i, y_i)\}$, chaque données x_i est caractérisée par p attributs et par sa classe y_i.
On cherche une hypothèse H telle que : H satisfait les échantillons $\forall\ x_i \in 1,\ldots,$ n H $(x_i)= y_i$
La classification consiste donc, en s'appuyant sur l'ensemble d'exemples à prédire la classe de toute nouvelle donnée x∈D [54].

1.3.2 Méthodes de classification

1.3.2.1 Réseaux de neurones

Un réseau de neurones est un modèle de calcul dont le fonctionnement vise à simuler le fonctionnement des neurones biologiques, il est constitué d'un grand nombre d'unités (neurones) ayant chacune une petite mémoire locale et interconnectées par des canaux de communication qui transportent des données numériques. Ces unités peuvent uniquement agir sur leurs données locales et sur les entrées qu'elles reçoivent par leurs connections. Les réseaux de neurones sont capables de prédire de nouvelles observations (sur des variables spécifiques) à partir d'autres observations (soit les même ou d'autres variables) après avoir exécuté un processus d'apprentissage sur des données existantes [50].

La phase d'apprentissage d'un réseau de neurones est un processus itératif permettant de régler les poids du réseau pour optimiser la prédiction des échantillons de données sur lesquelles l'apprentissage été fait. Après la phase d'apprentissage le réseau de neurones devient capable de généraliser [50].

Avantages :
Modélisation probabiliste de l'incertitude, et mise en ouvre de solution.
Les réseaux de neurones sont théoriquement capables d'approximer n'importe quelle fonction continue et ainsi le chercheur n'a pas besoin d'avoir aucunes hypothèses du modèle sous-jacent [50].

Inconvénients :

- Généralement les réseaux de neurones ne sont pas souvent utilisées dans les tâches du datamining parce qu'ils produisent des modèles souvent incompréhensibles et demande un longtemps d'apprentissage.
- Couteuses en temps d'exécution.
- Calcul explicite de probabilités sur des hypothèses.
- Approche pratique pour certains types de problèmes d'apprentissage.
- Des connaissances a priori peuvent être combinées avec les données observées [50].

1.3.2.2 Machines à vecteurs de support

Les Machine à Vecteurs de Support (SVM) ont été introduites par [57], Le SVM revient à chercher un hyperplan dont la distance entre les exemples d'apprentissage positifs et négatifs est maximale. L'hyperplan optimal séparant les points de deux classes est celui qui passe « au milieu » de ces classes, c'est-à-dire dont la distance aux points les plus proches est maximale.
Ces exemples les plus proches qui suffisent à déterminer cet hyperplan sont appelés vecteurs de support,

ou encore exemples critiques.
La distance séparant l'hyperplan de ces points est appelée « marge » [57].

Avantages :

- Fondamentaux mathématiques solides.
- Les exemples de tests sont comparés juste avec les supports vecteurs.
- Décision rapide pour la classification de nouveaux exemples [57].

Inconvénients :

- Classification binaire donc on ne peut faire que du (un contre un).
- Grand quantité d'exemple en entrée, calcul matriciel important.
- Temps de calcul élevé lors de régularisation des paramètres de la fonction noyau.[57]

1.3.2.3 Arbres de décision

Les arbres de décisions sont des outils d'aide à la décision qui permettent selon des variables discriminantes de répartir une population d'individus en groupes homogènes en fonction d'un objectif connu. Les arbres de décision sont des outils puissants et populaires pour la classification et la prédiction. Un arbre de décision permet à partir des données connues sur le problème de donner des prédictions par réduction, niveau par niveau, du domaine des solutions [8].

Avantages :

- les arbres de décision sont capables de produire des règles compréhensibles.
- Les arbres de décision effectuent la classification sans exiger beaucoup de calcul.
- Les arbres de décision sont en mesure de manipuler à la fois les variables continues et catégorielles [8].

Inconvénients :

- Manque de performance dans le cas de plusieurs classes ; les arbres deviennent très complexes et ne sont pas nécessairement optimaux.
- Demande beaucoup de temps de calcul lors de la construction (le choix du meilleur partitionnement) et l'élagage (la comparaison de sous-arbres).
- Moins bonnes performances concernant les prédictions portant sur des valeurs numériques.[8]

1.3.2.4 Réseaux Bayésiens

Modèle graphique qui encode les probabilités entre les variables plus pertinentes associer une probabilité d'apparition d'un événement étant donné la connaissance d'autres événements comprendre certaines relations causales notion d'antériorité ou d'impact), et aussi une conjonction de certaines variables pour déclencher une action il utilise l'ensemble des algorithmes basé sur la loi de Bayes, propose d'une part, que la connaissance sur le monde soit traduite par un ensemble d'hypothèse (fini ou non), chacune d'entre-elles étant affectée d'une probabilité reflétant le degré de croyance de l'apprenant dans l'hypothèse en question, la connaissance sur le monde est ainsi exprimée sous la forme d'une distribution de probabilités sur un espace d'hypothèses [39].

L'apprentissage consiste alors à trouver une structure de dépendances entre variables ou à estimer les probabilités conditionnelles définissant ces dépendances. Ainsi, les modèles basés sur cette formule, permettent d'exprimer des relations probabilistes entre les ensembles de faits, le raisonnement adopté dans ce cas est conditionnel, deux faits peuvent être en relation causale [39].

Avantages :

- Modélisation probabiliste de l'incertitude.
- Mise en ouvre de solution [46].

Inconvénients :

- Coûteuses en temps d'exécution.
- Calcul explicite de probabilités sur des hypothèses.
- Approche pratique pour certains types de problèmes d'apprentissage.
- Des connaissances a priori peuvent être combinées avec les données observées [46].

1.3.2.5 Classifieurs à base d'exemples

Les Classifieurs à base d'exemples n'établissent pas une représentation déclarative explicite des catégories mais se fondent sur le calcul directe de la similarité entre les données à classifier et les données d'apprentissage, la phase d'apprentissage est réduite à un stockage des exemples d'apprentissage.
Le classifier à base d'exemples le plus connu est les k-plus proches voisins (k-Nearest Neighbors), il procède par la prédiction de classe d'un individu x en fonction des k individus les plus proches voisins déjà étiquetés en mémoire.
L'emploi de k voisins, au lieu d'un seul, assure une plus grande robustesse à la prédiction.
Toutefois, la valeur de k peut changer les performances du modèle [64].
Cette méthode dépend des trois éléments suivants :

1 Le nombre de voisins retenus.

2 La mesure de distance entre exemple.

3 La combinaison des classes [64].

Avantages :

- Apprentissage rapide.
- Adapté aux domaines où chaque classe est représentée par plusieurs prototypes et ou les frontières sont irrégulières [64].

Inconvénients :

- Prédiction lente car il faut avoir tous les exemples à chaque fois.
- Sensible aux attributs non pertinents et corrélés [64].

1.4 Conclusion

Nous venons de présenter la fouille de données comme une technologie nouvelle et puissante consiste à extraire des informations cachés dans des grandes bases de données, ainsi que la classification de données en tant que tâche principale dans la fouille de données.

Chapitre 2

Sélection d'attributs : Etat de l'art

2.1 Introduction

Il est nécessaire lors de la construction d'un modèle de classification dans le domaine de fouille de données, de limiter et réduit le nombre d'attributs pris en compte, de manière à optimiser ses performances [48].
Les chercheurs rendent compte que la sélection d'attributs est l'une des techniques fréquentes utilisée dans l'étape de prétraitement de données pour que la fouille de données atteigne ses objectifs [26].
Dans ce chapitre, nous allons présenter le problème de sélection d'attributs ainsi que les différentes méthodes proposées dans la littérature pour résoudre ce problème.

2.2 Sélection d'attributs

2.2.1 Définition formelle

Selon les auteurs de [63] et [1], la sélection d'attributs pour la classification des données est généralement définie comme un processus de recherche dans un espace d'hypothèses (appelé ensemble de solutions possibles) permettant de trouver un sous-ensemble "pertinent" d'attributs parmi celles de l'ensemble de départ.
La notion de pertinence d'un sous-ensemble d'attributs dépend toujours des objectifs et des critères du modèle de classification construit [48].
Formellement le problème de sélection d'attributs peut être défini par [45] :
Soit A = $\{a_1, a_2, \ldots, a_D\}$ un ensemble d'attributs de taille D où D représente le nombre total d'attributs étudiés.
Soit F une fonction qui permet d'évaluer un sous-ensemble d'attributs, nous supposons que la plus grande valeur de F soit obtenue pour le meilleur sous-ensemble d'attributs.
L'objectif de la sélection est de trouver un sous-ensemble $A\prime$ ($A\prime \subseteq$ A) de taille $D\prime$ ($D\prime <=$ D) tel que :

$$\mathrm{F}(A\prime) \geq \mathrm{F}(\mathrm{A}) \text{ et } D\prime < \mathrm{D}$$

La recherche d'un sous-ensemble d'attributs optimal pour un critère que l'on s'est donné fait parti des problèmes d'optimisation NP-difficiles .[68][12]
Plusieurs approches peuvent être envisagées pour contourner cette difficulté(**Voir la section 2.3**).

2.2.2 Processus général de sélection d'attributs

Toutes les méthodes proposées dans la littérature pour la sélection d'attributs peuvent être décrites par un schéma général proposé par [48], basé sur quatre étapes :

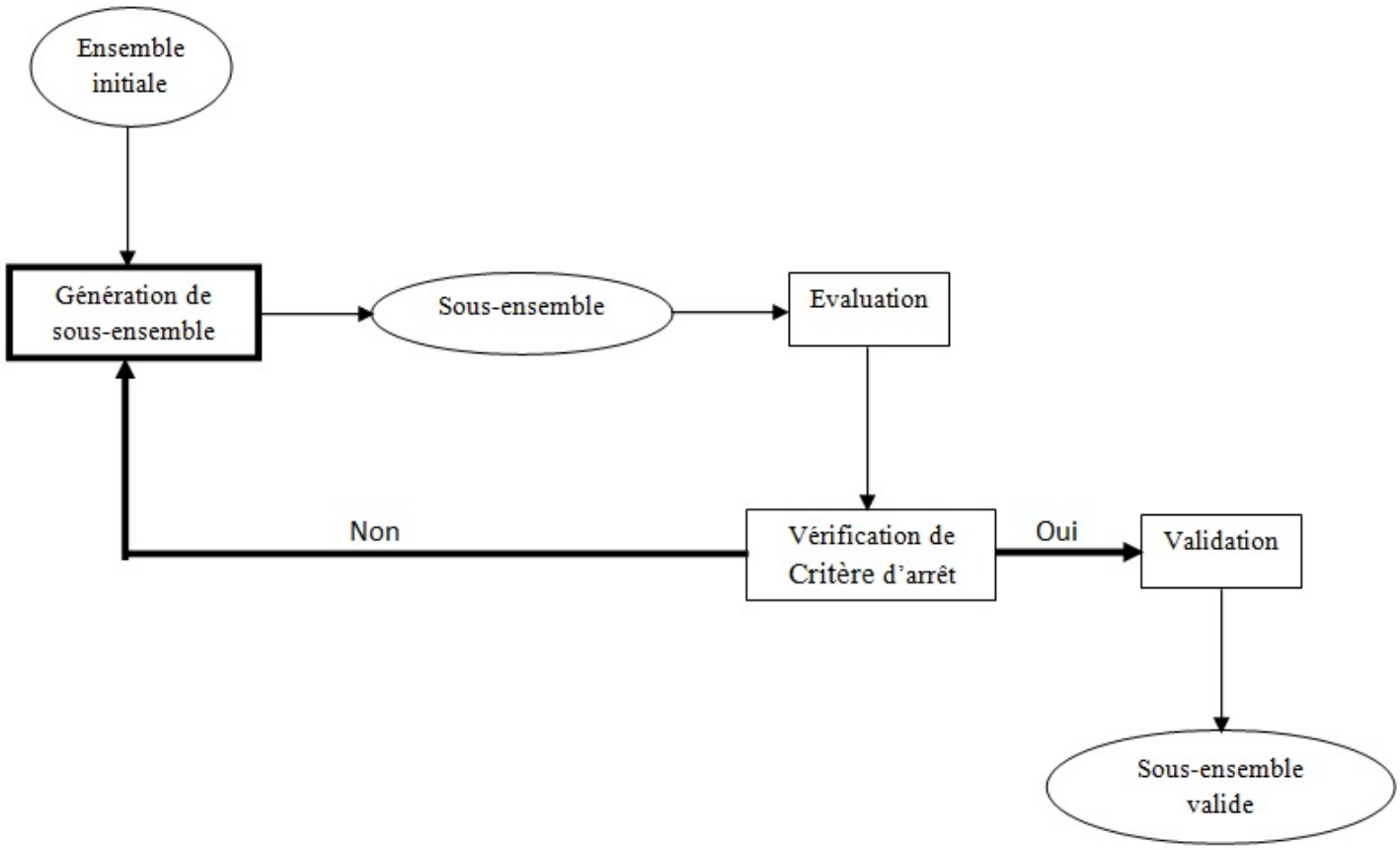

FIGURE 2.1 – Processus général de sélection d'attributs.

2.2.2.1 Génération de sous-ensemble

La génération de sous-ensemble d'attributs c'est une procédure de recherche, permettant d'explorer l'espace de recherche pour construire les déférentes combinaisons d'attributs.
Étant donné un ensemble initial X de n attributs, la sélection d'un "bon" sous-ensemble d'attributs nécessite d'examiner potentiellement 2^n-1 sous-ensembles possibles.
Cette procédure produit des sous-ensembles d'attributs candidats suivant une certaine stratégie de recherche. Par conséquent, différentes stratégies ont été explorées : recherche complète, séquentielle (heuristique) et aléatoire. [10] **(Voir la section 2.3.1.1)**
Ces sous-ensembles seront par la suite évalués selon un critère bien déterminé.[10].

2.2.2.2 Évaluation

Chaque nouveau sous-ensemble produit doit être évalué par un certain critère d'évaluation.
L'évaluation de sous-ensemble généré par la procédure de génération à comme objectif de mesurer la capacité d'un ensemble d'attributs à discriminer les classes de la partition impliquée par l'attribut endogène. [22]
Par exemple : Dans le cas d'un problème de classification supervisée, ce critère est très souvent la précision d'un classifieur construit à partir de l'ensemble d'attributs sélectionnés.
L'optimalité d'un sous-ensemble est relative à la fonction d'évaluation utilisée, un meilleur sous-ensemble est déterminé par certains critères (peut être optimal pour un critère donné et pas pour d'autres).[10]

2.2.2.3 Critère d'arrêt

Certains critères doivent être définis pour arrêter le processus de recherche sur les sous-ensembles d'attributs.
Le critère d'arrêt peut être un temps de calcul fixé, un nombre d'itérations fixé, ou encore le fait que les sous-ensembles candidats deviennent trop homogènes, etc [10].

2.2.2.4 Validation

Cette étape consiste à vérifier si le sous-ensemble est valide ou non, en d'autre terme si l'objectif souhaité est atteint ou non.
Une manière de faire la validation des résultats est de mesurer directement ces derniers en utilisant des connaissances a priori sur les données.
Par exemple, si nous employons le taux d'erreur de classification comme un indicateur de performance pour le traitement, pour un sous-ensemble d'attributs sélectionné, nous pouvons simplement suivre l'expérience "avant et après" pour comparer le taux d'erreur de classificateur sur l'ensemble complet d'attributs et sur le sous-ensemble sélectionné. [48]

2.2.3 Pertinence et non pertinence d'un attribut

La performance d'un algorithme de la fouille de données dépend fortement des caractéristiques (Attributs) redondantes ou non pertinentes, ce qui peut réduire cette performance [13].
La notion de pertinence, n'a pas de définition formelle qui est acceptée par tous [13], Dans la littérature il existe plusieurs définitions de la pertinence d'une caractéristique, la plus connue est :

- Celle de [22] et [21], selon cette définition, une caractéristique est classée comme étant très pertinente, peu pertinente et non pertinente.

  Très pertinente : Une caractéristique A_i est dite très pertinente si son absence entraine une détérioration significative de la performance du système de classification utilisée.

  Peu pertinente : Une caractéristique A_i est dite peu pertinente si elle n'est pas "très pertinente" et s'il existe un sous-ensemble V tel que la performance de V $\cup$ A_i soit significativement meilleure que la performance de V.

  Non pertinente : Les caractéristiques qui ne sont ni "peu pertinentes" ni "très pertinentes" représentent les caractéristiques non pertinentes. Ces caractéristiques seront en général supprimées de l'ensemble de départ.
- D'après [71] Une caractéristique pertinente est une caractéristique telle que sa suppression entraine une détermination des performances du pouvoir de discrimination en classement ou la qualité de prédiction en régression du système d'apprentissage.

2.3 Méthodes de sélection d'attributs

Dans la littérature, différentes méthodes ont été développées et utilisées pour la sélection d'attributs, On distingue les méthodes classiques et les métaheuristiques.

2.3.1 Méthodes classiques

Les auteurs de [10] proposent plusieurs catégorisations de ces méthodes selon deux critères : les stratégies de recherches et les critères d'évaluation [30] :

2.3.1.1 Catégorisation selon la Stratégie de recherche

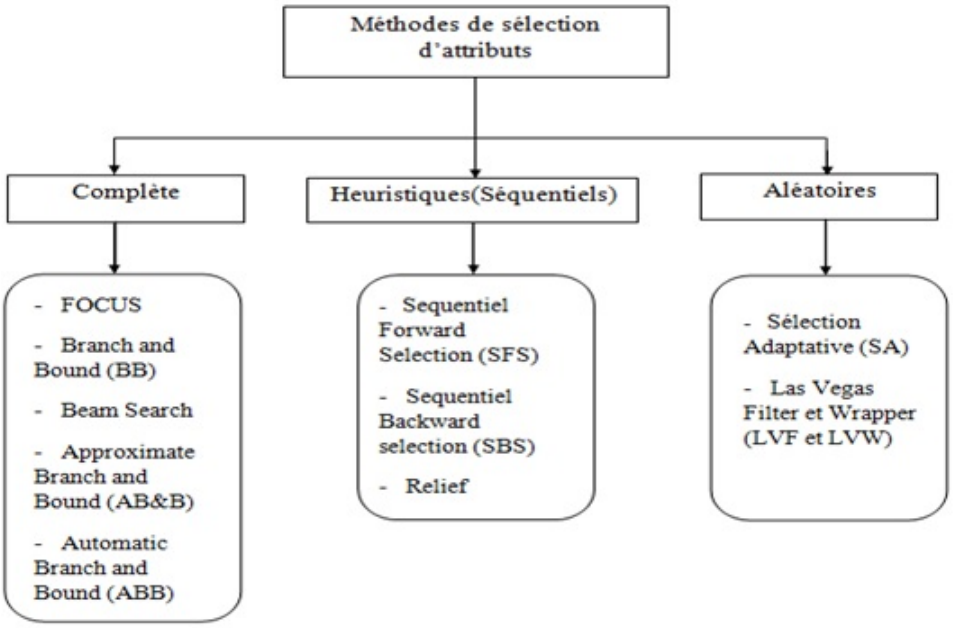

FIGURE 2.2 – Méthodes classiques de sélection selon les stratégies de recherche.

1. **Méthodes complètes :**
   Les méthodes regroupées dans cette catégorie effectuent une recherche complète du sous-ensemble optimal par rapport à la fonction d'évaluation choisie [48]. Cette recherche consiste à examiner toutes les combinaisons d'attributs possibles, ces méthodes ont une complexité de calcul très élevée (espace de recherche de l'ordre de O (2^N) pour N attributs) mais elles garantissent de trouver le sous-ensemble optimal d'attributs [48][53]. L'optimalité du sous-ensemble d'attributs par rapport au critère d'évaluation utilisée est garantie, car ces méthodes utilisent une procédure de Backtracking (retours arrière) qui peut être réalisée en utilisant des techniques telles que : FOCUS, Branch and Bound (BB), Best First Search (BFF), Beam Search, Approximate Branch and Bound (AB & B), Automatic Branch and Bound (ABB) [24].

2. **Méthodes séquentielles (heuristiques) :**
   Cette catégorie regroupe les algorithmes itératifs pour lesquels chaque itération permet de sélectionner, rejeter ou éliminer un ou plusieurs attributs, la recherche s'effectue de sorte qu'il ne soit pas nécessaire d'évaluer l'ensemble de tous les sous-ensembles d'attributs possibles, c'est pour cela que les méthodes heuristiques ne garantissent pas de trouver le sous-ensemble optimal. Parmi les techniques utilisées, on trouve : Sequential Forward sélection (SFS), Sequential Backward sélection (SBS), Relief [52].

3. **Méthodes aléatoires :**
   Elles commencent avec un sous-ensemble sélectionné aléatoirement et procède de deux façons différentes.
   (a) Consiste à continuer la génération des sous-ensembles avec la recherche séquentielle (type I)

   (b) Consiste à générer le sous-ensemble suivant une manière complètement aléatoire (type II) (c'est-à-dire le sous-ensemble courant n'est pas issu d'une augmentation ou diminution d'attributs du sous-ensemble précédent) [66].

Les résultats obtenus en utilisant ces types de méthodes dépendent du nombre des itérations, sans garantir que le sous-ensemble optimal soit atteint [10].
Parmi les techniques utilisées, on peut citer : Sélection Adaptative (SA), Las Vegas Filter et Wrapper (LVF et LVW) [66].

2.3.1.2 Catégorisation selon le critère d'évaluation

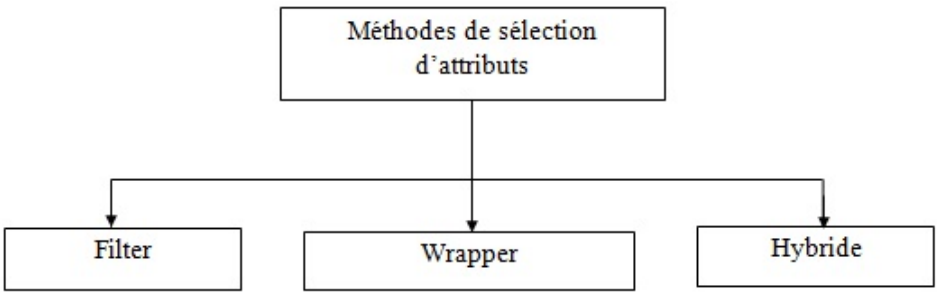

FIGURE 2.3 – Méthodes classiques de sélection selon les critères d'évaluation.

1. **Méthodes filtrantes :**
   Ces méthodes sont réalisées comme un prétraitement, elles sélectionnent les attributs en utilisant les différentes approches et les différents critères pour calculer la pertinence d'un attribut avant le processus d'apprentissage c'est-à-dire la construction d'un classifieur [13][72].
   Elles se basent sur la performance de la fonction d'évaluation calculée directement sur l'ensemble d'apprentissage comme : la distance, l'information, la dépendance, et la cohérence, sans tenir compte du classificateur [59].

2. **Méthodes Wrappers :**
   Ces méthodes consistent à générer des sous-ensembles candidats et de les évaluer grâce à l'algorithme d'induction, les méthodes Wrappers utilisent l'algorithme d'apprentissage comme une fonction d'évaluation, l'apprentissage est effectué avec les variables sélectionnées et les performances sont estimées à partir de l'erreur de généralisation.
   Les méthodes filtrantes sont plus rapides que les méthodes Wrappers en termes de génération de résultats. Cependant, ces dernières ont l'avantage de fournir généralement des résultats plus pertinents pour la classification.
   Ces méthodes nécessitent :
   - un espace d'états où chaque état représente un sous-ensemble d'attributs.
   - un état initial.
   - une condition d'arrêt.
   - une stratégie de recherche [23].

3. **Méthodes hybrides :**
   Les méthodes utilisent la mesure indépendante et l'algorithme d'induction pour évaluer les sous-ensembles d'attributs.
   Ces algorithmes se concentrent principalement sur la combinaison des algorithmes Filtre et Wrapper pour obtenir les meilleures performances possibles avec un algorithme d'apprentissage particulier.
   Il utilise la mesure indépendante pour décider le meilleur sous-ensemble pour une cardinalité donnée et utilise l'algorithme d'induction pour sélectionner le sous-ensemble final qui est le meilleure parmi les meilleures sous-ensembles à travers les différentes cardinalités [66].

Ces méthodes classiques sont généralement appliquées à des problèmes de taille réduite, c'est-à-dire que le nombre d'attributs de départ est réduit, pour un nombre supérieur, il devient un problème d'optimisation NB-difficile, il semble que peu d'algorithmes soient efficaces, pour cela les métaheuristiques sont apparues pour résoudre ce problème.

2.3.2 Métaheuristiques

Plusieurs définitions ont été proposées pour expliquer clairement la notion de métaheuristiques, nous citons parmi elles :
« Un processus itératif qui subordonne et qui guide une heuristique, en combinant intelligemment plusieurs concepts pour explorer et exploiter tout l'espace de recherche. Des stratégies d'apprentissage sont utilisées pour structurer l'information afin de trouver efficacement des solutions optimales, ou presque optimales » [29].
« Les métaheuristiques sont généralement des algorithmes stochastique itératifs, qui progressent vers un optimum global, c'est-à-dire l'extremum global d'une fonction objectif » [4].

2.3.2.1 Catégorisation des métaheuristiques

Il existe plusieurs façons de classer les méthodes métaheuristiques. Selon les caractéristiques choisies, Nous allons nous appuyer sur la catégorisation qui distingue les métaheuristiques basés sur population unique (qui exploitent séquentiellement un seul voisinage) et les métaheuristiques basées sur des populations de solutions (qui exploitent plusieurs à la fois) [4].

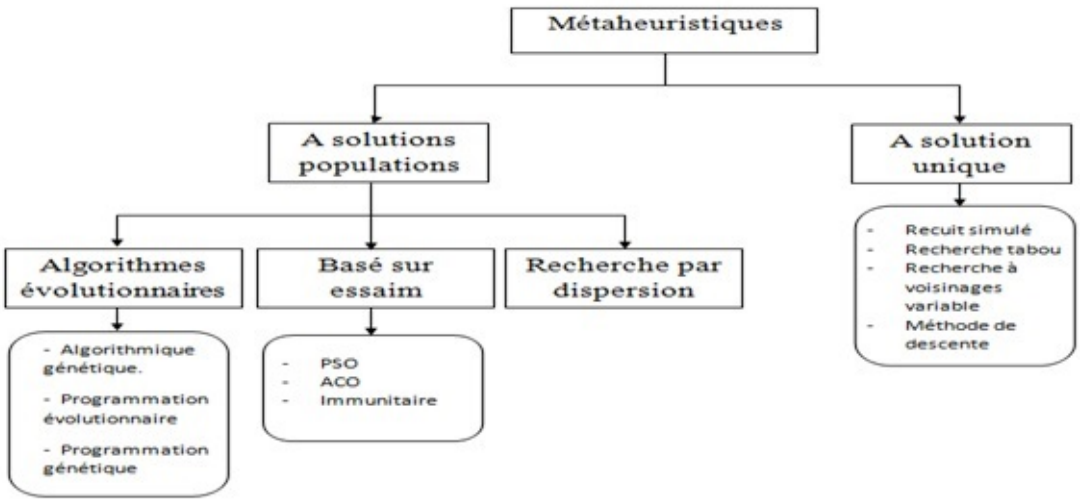

FIGURE 2.4 – Catégorisation des métaheuristiques pour la sélection d'attributs.

I). **Métaheuristiques à bases de population unique :**
Les méthodes à solution unique, plus connues sous le nom de méthodes de recherche locale, sont basées sur l'évolution d'une seule solution dans l'espace de recherche, utilisent la notion du voisinage pour améliorer la qualité de la solution courante.
Débutent la recherche avec une seule solution initiale, puis à chaque itération, la solution courante est déplacée dans un voisinage local en espérant améliorer une fonction objectif [9][10].

(a) **Méthode de Descente (recherche locale) :**
La méthode de descente représente la technique la plus intuitive et la plus simple, leur sucée revient à leur simplicité et à leur rapidité. Cette méthode fonctionne selon le principe suivant : en démarrant d'une solution aléatoire, à chaque itération du processus d'amélioration, l'algorithme modifie un ensemble de composantes de la solution courante pour

permettre le déplacement vers une solution voisine de meilleure qualité [9].
Le critère d'arrêt est atteint lorsque plus aucune solution voisine n'améliore la solution courante (optimum local est atteint) [10].
Le principal inconvénient de cette méthode est qu'elle reste piégée dans le premier optimum local Rencontré [10].

Algorithm 1 La recherche descente

```
Construire une solution initiale s ;
Calculer la fitness f(s) de s ;
Tant que la condition d'arrêt n'est pas vérifier faire
    Modifier s pour obtenir une nouvelle solution voisine s' ;
    Calculer f(s') ;
    si f(s') est meilleure que f(s) alors
        Remplacer s par s' ;
    Fin si
Fin tant que
Retourner s ;
```

(b) **Recherche à Voisinages Variables (VNS) :**
Méthode proposée par [56], A l' opposé d'autres méthodes basées sur la recherche locale, VNS utilise plusieurs voisinages dans un ordre prédéfini pour exploiter la solution courante. Etant donné un ensemble de structures de voisinages, une solution est générée aléatoirement dans le voisinage de la solution courante dont laquelle une méthode de descente locale est effectuée. Si l'optimum local obtenu n'est pas le meilleur que la solution courante, on passe vers le voisin suivant (répétition de procédure de recherche). On recommence la recherche à partir du premier voisinage après chaque amélioration, où il y a d'autres solutions meilleures qui n'ont pas été explorées ou on explore chaque structure de voisinage.
L'idée de parcourir un ensemble de voisinage permet de sauver la recherche du blocage causé par le piège de l'optimum local. Cette stratégie permet de diversifier l'exploration de l'espace de recherche et donne au processus de recherche plus de chances de rencontrer l'optimum global [56].

Algorithm 2 La recherche à voisinage variable

```
Définir la structure du voisinage Nk où k {1, 2, ..., k_max} ;
Construire une solution initiale s ;
Calculer la fitness f(s) ;
Tant que la condition d'Arrêt n'est pas vérifier faire
    k=1 ;
    Tant que k ≤ k_max faire
        Générer une solution s', où s' ∈ s
        Calculer f(s') ;
        si f(s) est meilleure que f(s') alors
            s=s' ;
            k=1 ;
        else
            k=k+1 ;
        Fin si
    Fin tant que
Fin tant que
Retourner s ;
```

(c) **Recuit Simulé (Simulated Annealing)**
La méthode du recuit simulé proposée par Kirkpatrick, Gelatt et Vecchi [5], a été inspirée de la technique de simulation de Metropolis [55] en mécanique statistique basée sur la distribution de Boltzmann.
Par analogie avec le processus physique, la fonction à minimiser est l'énergie du système E, on utilise aussi un paramètre « la température du système T ».
A partir de l'espace de solution, s0 est choisi aléatoirement, on associe avec cette solution une énergie initiale E=E0 et une température initiale T=T0 élevée.
Le principe de cette méthode est le suivant : une configuration (solution) initiale est générée aléatoirement, puis à chaque itération la solution courante est déplacée de façon aléatoire dans son voisinage local, générant ainsi une nouvelle solution. Si cette nouvelle solution est meilleure que la précédente, elle est retenue. Sinon elle est acceptée avec une certaine probabilité. Cette étape est réitérée tant que l'équilibre thermodynamique n'est pas atteint. Une fois l'équilibre atteint, la température est diminuée jusqu'à un nouvel équilibre. Et ainsi de suite, jusqu'à une condition d'arrêt (nombre d'itérations maximal atteint) [5].

Algorithm 3 Recuit simulé

```
Déterminer une configuration aléatoire S ;
Choix des mécanismes de perturbation d'une configuration ;
Initialiser la température T ;
Tant que la condition d'Arrêt n'est pas vérifier faire
    Tant que l'équilibre n'est pas atteint faire
        Tirer une nouvelle configuration S' ;
        Appliquer la régle de Metropolis ;
        si f(s) ≥ f(S') alors
            S_min=S' ;
            f_min=f(S') ;
        Fin si
    Fin tant que
    Décroitre la température ;
Fin tant que
```

(d) **Recherche Taboue (Tabu Search) :**
Est une méthode de recherche locale avancée, elle fait appel à un ensemble de règles et de mécanismes généraux pour guider la recherche de manière intelligente.
Le principe de la méthode est le suivant : à chaque itération, le voisinage de la solution courante est examiné et la meilleure solution est sélectionnée.
La méthode autorise de remonter vers des solutions qui semblent moins intéressantes mais qui ont peut être un meilleur voisinage.
Pour éviter les phénomènes de cyclage (cycler entre deux solutions), la méthode a l'interdiction de visiter une solution récemment visitée. Pour cela, une liste taboue contenant les attributs des dernières solutions visitées est tenue à jour.
Chaque nouvelle solution considérée enlève de cette liste la solution la plus anciennement visitée.
Ainsi, la recherche de la solution courante suivante se fait dans le voisinage de la solution courante actuelle sans considérer les solutions appartenant à la liste taboue.
La méthode de recherche taboue présente l'avantage de comporter moins de paramètres que l'algorithme de recuit simulé [5].

Algorithm 4 La recherche taboue

Construire une solution initiale s ;
Calculer la fitness f(s) ;
S_{best}=S ;
Tant que la condition d'Arrêt n'est pas vérifier **faire**
Trouver la meilleure solution $S\prime$ dans le voisinage de s qui ne soit pas tabou ;
Calculer f($S\prime$) ;
si $f(S\prime) \geq f(S_{best})$ **alors**
S_{best}=$S\prime$;
Fin si
Mettre à jour la liste taboue ;
S=$S\prime$;
Fin tant que Retourner S_{best} ;

II). **Métaheuristiques à bases de population de solutions :**
Les métaheuristiques à population de solutions sont des méthodes qui font évoluer simultanément un ensemble d'individus (solutions) dans l'espace de recherche au fur et à mesure des itérations [4]. Ces méthodes sont principalement inspirées du vivant. On peut distinguer deux catégories de métaheuristiques à population : les algorithmes évolutionnaires inspirés de la théorie de l'évolution de C. Darwin [6] et les algorithmes d'intelligence en essaim inspirés de l'éthologie et de la biologie.[4]

(a) **Algorithmes Génétiques :**
Les algorithmes génétiques sont des stratégies d'adaptation et des techniques d'optimisation globale [2], qui s'appuient sur des techniques dérivées de la génétique et de l'évolution naturelle : sélection, croisement, mutation. Ils ont été inventés dans les années 60 [36]. L'auteur Goldberg a étudié les Algorithmes Génétiques et il a enrichit sa théorie par [14] :

— Un individu est lié à un environnement par son code d'ADN.
— Une solution est liée à un problème par son indice de qualité.
— Une bonne solution à un problème donné peut être vue comme un individu susceptible de survivre dans un environnement donné.

Un AG a généralement 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 [2].

Opérations des algorithmes génétiques : Le processus de recherche de l'algorithme génétique est fondé sur les opérateurs suivants [67] :

- **Un opérateur de codage des individus :** Il permet la représentation des chromosomes représentant les individus.
- **Un opérateur d'initialisation de la population :** Il permet la production des individus de la population initiale. Malgré que cet opérateur ne s'intervient qu'une seule fois et au début de la recherche, mais il joue un rôle non négligeable dans la convergence

vers l'optimum global. En fait, le choix de la population initiale peut rendre la recherche de la solution optimale du problème traité plus facile et plus rapide.

- **Un opérateur de sélection :** Il permet de favoriser la reproduction des individus qui ont les meilleures fitness (i.e. les meilleures qualités).
- **Un opérateur de croisement :** Il permet l'échange des gènes entre parents (deux parent en général), pour créer 1 ou deux enfants en essayant de combiner les bonnes caractéristiques des parents, consiste à choisir un seul point de coupure, puis échanger les fragments situés après ce point coupure, comme le montre la figure :

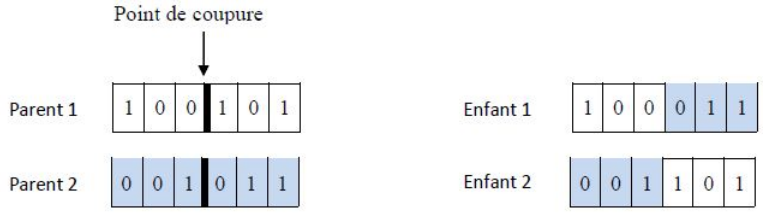

FIGURE 2.5 – Opération de croisement.

- **Un opérateur de mutation :** Il consiste à modifier quelques gènes des chromosomes des individus, dans le but d'intégrer plus de diversité au sein du processus de la recherche.

L'algorithme 5 montre le fonctionnement simplifié d'un algorithme génétique [67] :

Algorithm 5 Algorithme génétique

Initialiser les paramétres nécessaires ;
Initialiser une population de N individus ;
Evaluer les N individus ;
Tant que la condition d'Arrêt n'est pas vérifier **faire**
  Utiliser l'opérateur de sélection pour sélectionner K individus ;
  Appliquer l'opérateur de croisement sur K individus avec la probabilité P_c ;
  Appliquer l'opérateur de mutation sur les individus avec la probabilité P_m ;
  Utiliser l'opérateur d'évaluation pour évaluer les enfants obtenus ;
  Utiliser l'opérateur de sélection pour remplacer quelques individus parents par des individus enfants ;
Fin tant que
Retourner la ou les meilleures solutions ;

(b) **Le système immunitaire artificiel :**

Les systèmes immunitaires artificiels sont des modèles des métaheuristiques inspirées des systèmes immunitaires naturels.

Le système immunitaire biologique naturel constitue une arme contre des intrus dans un corps donné. Pour ce faire, il existe plusieurs cellules qui contribuent à éliminer ces intrus nommé antigènes. Ces cellules participent pour ce qu'on appelle une « réponse immunitaire biologique » [33], il est assez compliqué pour qu'une simulation artificielle soit réalisée d'une façon complète.

Par contre, les chercheurs ont réussi à simuler les fonctions les plus pertinentes dans un système immunitaire biologique pour que l'artificiel hérite le maximum des fonctionnalités naturelles dans le domaine de la reconnaissance des formes [15].

Le système immunitaire artificiel est très utile pour résoudre les problèmes de bioinformatique et de modélisation du processus biologique en appliquant des algorithmes immunitaires [48].

On distingue : l'algorithme de sélection clonale [3], l'algorithme de sélection négative et l'algorithme du réseau immunitaire [47], parmi ces algorithmes le plus utilisé est celui de sélection clonale.

Algorithme de sélection clonale :

— La sélection clonale est une abstraction des mécanismes de mémorisation des systèmes immunitaires, dont le problème à traiter joue le rôle d'un antigène.

— Les solutions possibles jouent le rôle des anticorps d'une cellule B nommée Lymphocyte B. Lorsque les anticorps d'une cellule B rencontrent un antigène, la cellule B commence à créer des clones (des copies exactes de la cellule B), ces derniers subissent des mutations afin d'améliorer leurs affinités en adaptant leurs anticorps aux antigènes envahissants.

Algorithm 6 Algorithme de sélection clonale

Initialiser une population de N anticorps ;
Calculer leurs affinités ;
Tant que la condition d'Arrêt n'est pas vérifier **faire**
    Produire des clones des cellules de bonnes affinités ;
    Muter les clones produits ;
    Sélectionner les cellules de meilleures affinités ;
    Remplacer des cellules de meilleures affinités par les cellules sélectionnées ;
Fin tant que
Retourner les meilleures solutions ;

(c) **Optimisation par Colonie de Fourmis :**
L'algorithme d'optimisation par colonies de fourmis est un des algorithmes basés sur l'intelligence par essaim, l'idée de base du trinôme imite le comportement collectif des fourmis lors de leur déplacement entre la fourmilière et la source de nourriture.
Dans ACO le problème de sélection d'attributs soit représenté comme un graphe.Ici, les nœuds représentent les caractéristiques, les arêtes entre eux indiquent le choix de la prochaine caractéristique, la recherche du sous-ensemble optimal de caractéristiques est alors un parcours des fourmis dans le graphe avec un nombre minimal de nœuds visités satisfaisant le critère d'arrêt du parcours. Phéromone associée à chaque caractéristique c'est une valeur heuristique associée à chaque attribut sélectionné [49].

Algorithm 7 Optimisation par colonie de fourmis

Définir les paramètres ;
Initialiser les traces de phéromones ;
Tant que la condition d'Arrêt n'est pas vérifier **faire**
    Construction des solutions par les fourmis ;
    Actions spécifiques (Facultatif) ;
    Sélectionner les cellules de meilleures affinités ;
    Mise à jour de phéromones ;
Fin tant que
Retourner les meilleures solutions trouvées ;

L'algorithme détaillé comme suit :

- Générer un certain nombre de fourmis k, qui sont ensuite placées au hasard sur le graphe (configuration aléatoire).
- Alternativement, le nombre de fourmis à placer sur le graphe peut être défini comme étant le nombre de caractéristique dans les données, chaque fourmi commence la construction du chemin à une caractéristique différente.
- A partir de ces postions initiale, elles parcourent les arêtes probabilistes jusqu'à ce qu'un critère d'arrêt de parcours soit satisfait.
- Les sous-ensembles résultats sont recueillis et évaluées.
- Si un sous-ensemble optimal à été trouver ou que l'algorithme à exécuté un certain nombre de fois, alors le processus s'arrête et donne le meilleur sous-ensemble d'attributs produits.
- Si aucune condition n'est vérifier alors la phéromone est crée et le processus réitère, une fois de plus [49].

(d) **Optimisation par Essaims particulaires :**
L'algorithme d'optimisation par essaims particulaires a été proposée par Kennedy et Eberhart en 1995 [42], cet algorithme inspiré du comportement social d'animaux évoluant en essaim.
La méthode PSO met en jeu un ensemble d'agents pour la résolution d'un problème donné, cet ensemble est appelé essaim, l'essaim est composée d'un ensemble de membres, ces derniers appelées particules. L'essaim de particules survole l'espace de recherche, en quête de l'optimum global, les particules représentent des solutions potentielles au problème traité.
Le déplacement de chaque particule dans l'espace de recherche est influencé par sa vitesse, la meilleure position qui a été retenue(Pbest) et la meilleure position connue par toutes les particules de l'essaim (Gbest).
L'optimisation par essaim particulaire est une jeune métaheuristique. Grâce à son efficacité, ses concepts simples et ca convergence rapide, elle a été utilisée pour la résolution de plusieurs problèmes dans différents domaines, parmis ces problème on distingue "La sélection d'attributs dans la classification des données" [42].

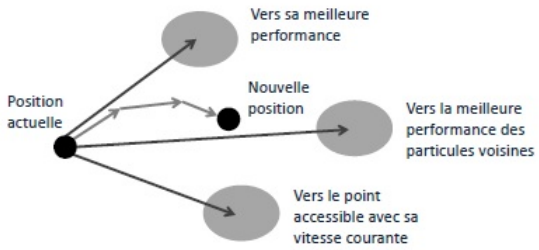

FIGURE 2.6 – Déplacement d'une particule.

Principe de l'algorithme PSO :

- Soit x_i un vecteur de position de la $i^{éme}$ particule de l'essaim.
- x_{id} représente le $d^{éme}$ bit de vecteur de position de particule i.
- v_i un vecteur de vitesse de cette particule.
- D la dimension de ce problème.

- Cette particule garde en mémoire la meilleure position par laquelle elle est déjà passée et la meilleure position atteinte par toutes les particules de l'essaim, notées respectivement : $Pbest_i$ et Gbest.
- La particule i va se déplacer entre les itérations t et t+1, en fonction de sa vitesse et des deux meilleures positions (Pbest, Gbest)suivant les deux équations suivantes :
  v_{id}(t+1) = v_{id} (t) + c_1 r_1 ($Pbest_{id}$(t) - x_{id} (t)) + c_2 r_2 (Gbest(t) - x_{id} (t)) (2.1)

  $$x_{id}(t+1) = x_{id}(t)+v_{id}(t+1) \quad (2.2)$$

  Avec :
  - x_i (t) et x_i (t+1) : position de particule i au temps t et t+1.
  - v_i (t) et v_i (t+1) : vitesse de particule i au temps t et t+1.
  - $Pbset_i$(t) : meilleure position obtenu par la particule i au temps t.
  - Gbest(t) : meilleure position obtenu par l'essaim au temps t.
  - c1, c2 : deux constantes qui représentent les coefficients d'accélération, elles peuvent être non constantes dans certains cas.
  - r1, r2 : nombres aléatoires tirés de l'intervalle [0,1].
- Afin d'estimer la qualité de la particule i, il est indispensable de calculer sa fonction «objectif» (aussi appelée fitness f), cette dernière est calculée en utilisant une fonction spéciale au problème traité [67].

- Pbest et Gbest met à jour est calculé suivant les equations :

$$Pbesti(t+1) = \begin{cases} xi(t+1) & \text{si } f(xi(t+1)) \geq f(Pbesti(t)) \\ Pbesti(t) & \text{sinon} \end{cases} \quad (2.3)$$

$$Gbest(t+1) = argmax_{Pbest_i} f(Pbest_i(t+1)), 1 \leq i \leq N \quad (2.4)$$

Algorithm 8 Optimisation par Essaim de Particules

Initialiser aléatoirement N particules : position et vitesse ;
for chaque particule i de N **faire**
  Calculer f(x_i) ;
  $Pbest_i$=x_i ;
Fin pour
Calculer Gbest selon (2.4) ;
Tant que le condition d'Arrêt n'est pas satisfait **faire**
  Déplacer les particules selon (2.1) et (2.2) ;
  Calculer f(x_i) pour chaque particule i ;
  Mettre à jour $Pbest_i$ pour chaque particule i selon (2.3) ;
  Mettre à jour Gbest selon (2.4) ;
Fin tant que Retourner Gbest ;

Cet algorithme a l'inconvénient de ne pas donner lieu à une exploration suffisante, ce qui peut conduire à une stagnation dans un optimum local et donc une convergence prématurée[67].

Variantes de l'algorithme PSO :
L'idée d'optimisation par essaim de particules a sollicité l'attention de plusieurs chercheurs qui ont mené des études dans l'objectif d'améliorer la performance de la méthode PSO proposée, on peut citer :

1) En 1996 Eberhart et ses collègues ont proposé de limiter la vitesse de la particule par l'intervalle [- vmax, +vmax],leur objectif était d'échapper au problème de déviation des particules de l'espace de recherche lors de leur déplacement, le nouveau paramètre vmax permet de mieux contrôler le mouvement de particules[62].
2) En 1998, Shi et Eberhart ont proposé dans [75] une variante de l'équation (2.1), la modification apportée consiste à appliquer un facteur d'inertie pour contrôler la vitesse des particules de la manière suivante :

$$v_{id}(t) = \omega v_{id}(t-1)+c_1 r_1(Pbest_{id}(t-1)-x_{id}(t-1))+c_2 r_2(Gbest(t-1)-x_{id}(t-1)) \quad (2.5)$$

Où ω est un coefficient d'inertie, généralement une constante qui sert à contrôler l'influence de la vitesse de la particule sur son prochain déplacement afin de garder un équilibre entre l'exploitation et l'exploration de l'espace de recherche, elle peut être variable dans certains cas : Eberhart et Shi voient que la valeur raisonnable de ω doit diminuer linéairement au cours du processus de l'optimisation [75].

3) De même Kusum Deep et Jagdish Chand Bansal proposent de réduire la valeur de ω linéairement de 0.8 à 0.4 [75].
4) De sa part, Clerc a proposé dans [73] une autre variante de l'équation (2.1), sa variante consiste à ajouter un facteur de constriction K dont l'objectif est de contrôler la vitesse des particules afin d'échapper au problème de la divergence de l'essaim qui cause la convergence prématurée de l'algorithme, l'équation permettant la mise à jour de la vitesse est la suivante :

$$v_{id}(t) = K[v_{id}(t-1)+c_1 r_1(Pbest_{id}(t-1)-x_{id}(t-1))+c_2 r_2(Gbest(t-1)-x_{id}(t-1))] \quad (2.6)$$

5) Au début, la métaheuristique d'optimisation par essaim de particules a été proposée pour résoudre des problèmes d'optimisation continus. Toutefois, beaucoup de problèmes d'optimisation sont discrets ou discrets/binaires. Par conséquent, une adaptation de l'algorithme PSO a été proposé pour résoudre des problèmes d'optimisation discrets ou binaires [37].
Dans l'algorithme BPSO (Binary Particle Swarm Optimization), la position de la particule i est représentée par un ensemble de bit, la vitesse v_i de la particule i est calculée à partir de l'équation (2.1).
v_i est un ensemble de nombres réels qui doivent être transformés en un ensemble de probabilités en utilisant la fonction sigmoïde comme suit :

$$sig(v_{id}) = \frac{1}{1+e^{-v_{id}}} \quad (2.7)$$

Où $sig(v_{id})$ représente la probabilité du bit X_{id} de prendre la valeur 1.

En se basant sur la vitesse v_i de la particule i et sur la valeur de la probabilité Sig(v_{id}), la position de la particule x_i est calculée comme suit :

$$x_{id} = \begin{cases} 1 & \text{si } rand() < sig(v_{id}) \\ 0 & \text{sinon} \end{cases} \quad , rand() \in [0,1] \quad (2.8)$$

6) Dans cette dernière décennie l'algorithme d'optimisation par essaim de particules quantiques (QPSO : Quantum Particle Swarm Optimization) a construit une nouvelle tendance, cette tendance vise à combiner les principes de la mécanique quantique et l'algorithme d'optimisation par essaim de particules afin de donner naissance à une nouvelle classe de d'algorithme PSO pour résoudre différents problèmes d'optimisation.
Dans le modèle classique du PSO, la particule se déplace en tenant compte de sa position actuelle dans l'espace de recherche et sa vitesse de déplacement, ce qui n'est pas le cas dans le modèle du QPSO où la vitesse n'a pas d'influence sur le déplacement de la particule [41].
En fait, dans la version quantique de l'algorithme PSO la particule se déplace selon les équations suisvantes :

$$M : \ Nombre\ de\ particules$$

$$mbest = \sum_{i=1}^{M} Pbest_i/M \tag{2.9}$$

$$x_{id}(t) = P_{id} \underline{+} \beta * |mbest_{id} - x_i(t)| * ln(1/u) \tag{2.10}$$

$$P_{id} = \varphi Pbest_{id} + (1-\varphi)Gbest \tag{2.11}$$

OÙ :
mbest : La meilleure position moyenne parmi les particules.
P_{id} : Un point stochastique entre $Pbest_{id}$ et Gbest, qui la dème coordonnée de la Attracteur local de la $i^{ème}$ particule p_i
β : est nommé coefficient de Contraction-Expansion.
u et φ : nombres tirés aléatoirement de l'intervalle [0,1].

7) Dans [41] BQPSO est la version discret de QPSO la position de la particule est représenté comme une chaine binaire :

$$Xi = (X_1, ..., X_D) \tag{2.12}$$

La distance entre deux chaine est représenté par un distance de Hamming, consiste à calculer le nombre de fois où les deux chaines sont différentes bit à bit, calculer par :

$$|X - Y| = d_H(X, Y) \tag{2.13}$$

ou bien :

$$d_H(X, Y) = b$$

avec :

$$b = d_H(X_i, P_i) = \beta(d_H(X_i, mbest))ln(1/u) \tag{2.14}$$

$$\beta, u : \ Nombres\ aléatoires$$

mbest calculer par la procédure 1 :

Get_mbest(Pbest)
Pour j=1 to d (d : taille de Pbest) **Faire**
sum=0 ;
  Pour tout particule i **Faire**
    sum=sum+Pbest[i][j] ;
  Finpour
  avg=sum/M ;
  Si avg $\gg$ 0.5 **Alors** mbest[j]=1 ;**Finsi**

Si avg $\ll$ 0.5 **Alors** mbest[j]=0 ;**Finsi**
Si avg=0.5 **Alors**
Si rand( ) $\ll$ 0.5 **Alors** mbest[j]=0 ;
Sinon mbest[j]=1 ;
Finsi
Finsi
Finpour
Retourner mbest ;

P_i obtenu par une opération de croisement entre Pbesti et Gbest, calculer par la procédure 2 :

Get_P($Pbest_i$**, Gbest)**
Appliquer une opération de croisement
entre $Pbest_i$, Gbest
pour générer deux vecteur binaire $z_1 et z_2$;
Si rand( )$\ll$0.5 **Alors** $P_i = z_1$;
Sinon $P_i = z_2$;
Finsi
Retourner p_i ;

La mise à jour de X_i obtenu par une mutation de P_i dans la procédure 3 (**Transf**) avec la probabilité Pr :

$$Pr = \begin{cases} 1 & \text{si } b/l \gg 1 \quad , l; taille\ de\ vecteur\ position \\ b/l & \text{sinon} \end{cases} \tag{2.15}$$

Transf(P_i**,Pr)**
Pour touts bit de Pi **faire**
Si rand()$\ll$Pr **Alors**
Si etat de bit est 1 alors remise à 0 ;
Sinon remise à 1 ;
Finsi
Finsi
Finpour
X_i=P_i ;
Retourner X_i ;

2.3.2.2 Hybridations basées sur optimisation par essaim

Le Tableau 2.1 ci-dessous montre quelques hybridations trouvées dans la littérature entre quelques métaheuristiques, basés sur l'optimisation par essaim de particules (PSO) :

Approche	**Principe**	**Auteures**
AG-PSO (HGPSO)	les individus d'une nouvelle génération sont crées par les opérations de croissement, mutation et PSO.	C-F.Juang [7]
PSO-AG	utiliser la logique floue pour intégrer les résultats des deux méthodes.	F.Valdez, P.Melin, O.Castillo [19]
PSO-AG	la mis à jour de position des meilleures particules faites par AG.	K.Premalatha, A.M.Natrajan [43]
AG-PSO	remplacer la dernière étape d'AG (mutation) par PSO.	H.Hachimi [25]
PSO-GA	introduit l'opération de sélection dans l'algorithme PSO pour choisir les meilleures particules et subir une mutation, ensuite les mauvais seront éliminées.	P.Angeline V [58]
PSO-GA	GA utilisé pour générer la population initiale de PSO.	Robinson [40]
AG-PSO	PSO utilisé pour générer la population initiale d'AG.	Robinson [40]
AG-PSO (GSO)	Population divisé en deux sous-populations. Suivie mecanisme AG et PSO pour chaque itération. Sous-populations seront regrouper pour actualiser la population initiale. Population initiale sera divisée à nouveau et recommencer l'algorithme.	E.A Grimaldi, F.Grimacia, M.Mussetta, P.Pirinoli, R.Zich [20]
AG-PSO (GSO)	inspiré de la première version de [20] avec introduction un nouveau paramètre constante d'hybridation qui indique le pourcentage de participation d'une population dans un AG pour chaque itération.	E.A Grimaldi, F.Grimacia, M.Mussetta, P.Pirinoli, R.Zich [20]
PSO-AG (NPSO)	intégrer les opérations (croissement, mutation) dans les étapes de la version standard PSO.	Lian [76]
PSO-AG	PSO avec l'opération de mutation.	Zhang [74]
PSO1	PSO avec coef K est dynamique, afin de maintenir la diversité de la population.	Clerc, kennedy [73]
PSO2	ré-implémenter l'algorithme de [73], proposé de générer la valeur de facteur de constriction K à partir de l'intervalle [0.2, 0.9] à chaque itération.	Achtning [31]
PSO-ACO	ACO appliquer pour effectuer une recherche locale dans laquelle les fourmis sont guidées par la concentration des phéromones pour mettre à jour les positions calculées par PSO.	P.S.Shelokar, V.K.Jayaramen [60]
PSO-ACO (PSACO)	ACO pour mettre à jour les particules de PSO.	P.S.Shelokar, V.K.Jayaramen [60]
PSO-ACO	ACO pour remplacer la meilleure position de chaque particule de PSO.	Habibi [34]
PSO-ACO-recuit simulé	ACO utilisé pour remplacer la position de chaque particule trouvée par PSO. -recuit simulé pour contrôler l'exploitation de meilleure particule du groupe.	Habibi [34]
PSO-VSN	PSO avec croissement, recherche locale à voisinage variable.	Czgolla, Fink [32]
PSO-SA	Au cours de l'algorithme, fait appel à SA, mutation, croissement.	X.Shen [70]
PSO-Recherche locale	PSO avec méthode de recherche locale afin d'améliorer la recherche localement, comme ils ont utilisé les concepts d'AG pour mieux explorer l'espace de recherche.	Lope, Coelho [28]

Tableau 2.1 – Hybridations entre métaheuristiques basés sur PSO

2.4 Conclusion

Dans ce chapitre, après avoir présenté la sélection d'attributs en tant que problème d'optimisation, et son importance pour améliorer la tâche de classification des données, nous avons décrit un état de l'art des méthodes de sélection et quelques hybridations entre eux proposées dans la littérature, en se concentrant sur les méthodes métaheuristiques.
Notre travail dans le chapitre suivant porte sur la proposition d'une nouvelle approche pour la sélection d'attributs.

Chapitre 3

Une approche BC-QPSO pour la sélection d'attributs

3.1 Introduction

Les métaheuristiques sont considérées comme étant une stratégie efficace pour la recherche des solutions approchées des problèmes d'optimisation NP-difficile, d'un autre coté plusieurs hybridations entre les métas heuristiques sont proposées dans la littérature pour résoudre le problème de sélection d'attributs dans la classification de données (la recherche de sous-ensemble d'attributs pertinents parmi la totalité des attributs d'une base de données).

Après notre étude, nous avons constaté l'absence des hybridations entre l'approche BQPSO (Binary Quantum Particle Swarm Optimization) et le système immunitaire artificiel pour la résolution du problème de sélection d'attributs, et pour cela notre travail consiste à proposer une approche hybride entre les PSO et l'algorithme de sélection clonale inspiré du système immunitaire artificiel.

Dans ce chapitre nous détaillons notre approche dénotée par BC-QPSO et nous présentons les résultats de tests et de comparaison avec d'autres approches de la littérature. Nous avons implémenté et testé notre approche sous Java et utilisant un logiciel de datamining « Weka ».

3.2 Conception de l'approche BC-QPSO proposée

3.2.1 Codage des particules

Chaque particule à une position, cette dernière est représentée par un vecteur binaire, les bits vaut 1 correspond aux attributs sélectionnés, et les bits vaut 0 aux attributs non sélectionnés.

Exemple : Soit la position de particule i représentée comme suit :

$$X_i = [1, 1, 0, 1, 0, 1]$$

Les éléments corresponds aux positions 1, 2, 4, 6 sont les attributs sélectionnées, alors que 3, 7 non sélectionnées.
La taille du vecteur correspond au nombre de dimensions (attributs dans la base de données).

3.2.2 Évaluation des particules

Nous avons utilisé pour évaluer les particules le taux de classification trouvé par la classification en appliquant l'algorithme K-ppv aux jeux de données, ce taux est considéré comme une fonction de fitness (fonction objectif).
On a utilisé dans nos expérimentations le classifieur K-ppv disponible dans Weka sous le nom (IBK).

3.2.3 Modélisation du problème de sélection des attributs par QBPSO

Le problème de sélection des attributs fait partie des problèmes d'optimisation discret / binaire, il est représenté dans l'algorithme BQPSO comme suit [41] :

- L'idée principale est d'utiliser un swarm de N particules, chaque particule i est représentée par sa position X_i qui représente une solution du problème.
- Chaque particule du swarm se déplace (mis à jour sa position) dans un espace appelé « l'espace de recherche ».
- A chaque position la particule construit une solution candidate dans un espace de recherche.
- Après qu'une particule change de position, une fonction d'évaluation de position (fonction fitness) est calculée c'est lui qui va évaluer la solution trouvée f(X_i).
- Mètre à jour la meilleure solution trouvée pat toutes particules i individuellement ($Pbest_i$) et la meilleure position globale trouvée par le swarm (Gbest).
- Le processus de recherche continue à chercher la meilleure solution selon la fonction fitness, jusqu'à ce que le critère d'arrêt soit vérifié, si ce dernier est vérifié, on retient la meilleure solution.
- Dans chaque itération, la même procédure sera répétée.
- Lorsque le critère d'arrêt est vérifié, la meilleure solution (Gbest) est retournée.

La figure 3.1 présente le fonctionnement général de l'algorithme QBPSO [41] :

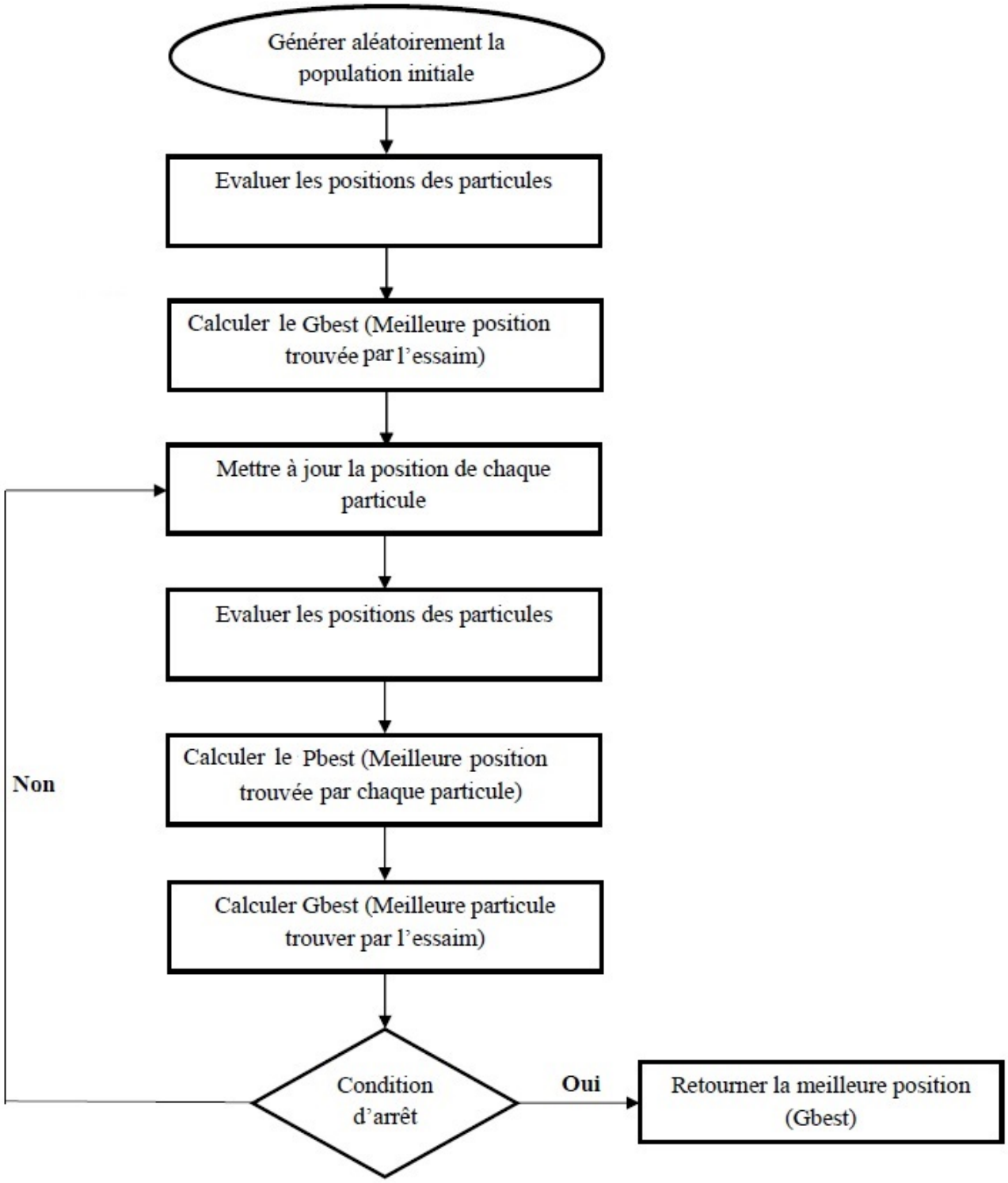

FIGURE 3.1 – Schéma fonctionnel de l'algorithme BQPSO.

3.2.4 L'algorithme BC-QPSO

La figure 3.2 ci-dessous présente le fonctionnement général de notre approche dénoté BC-QPSO :

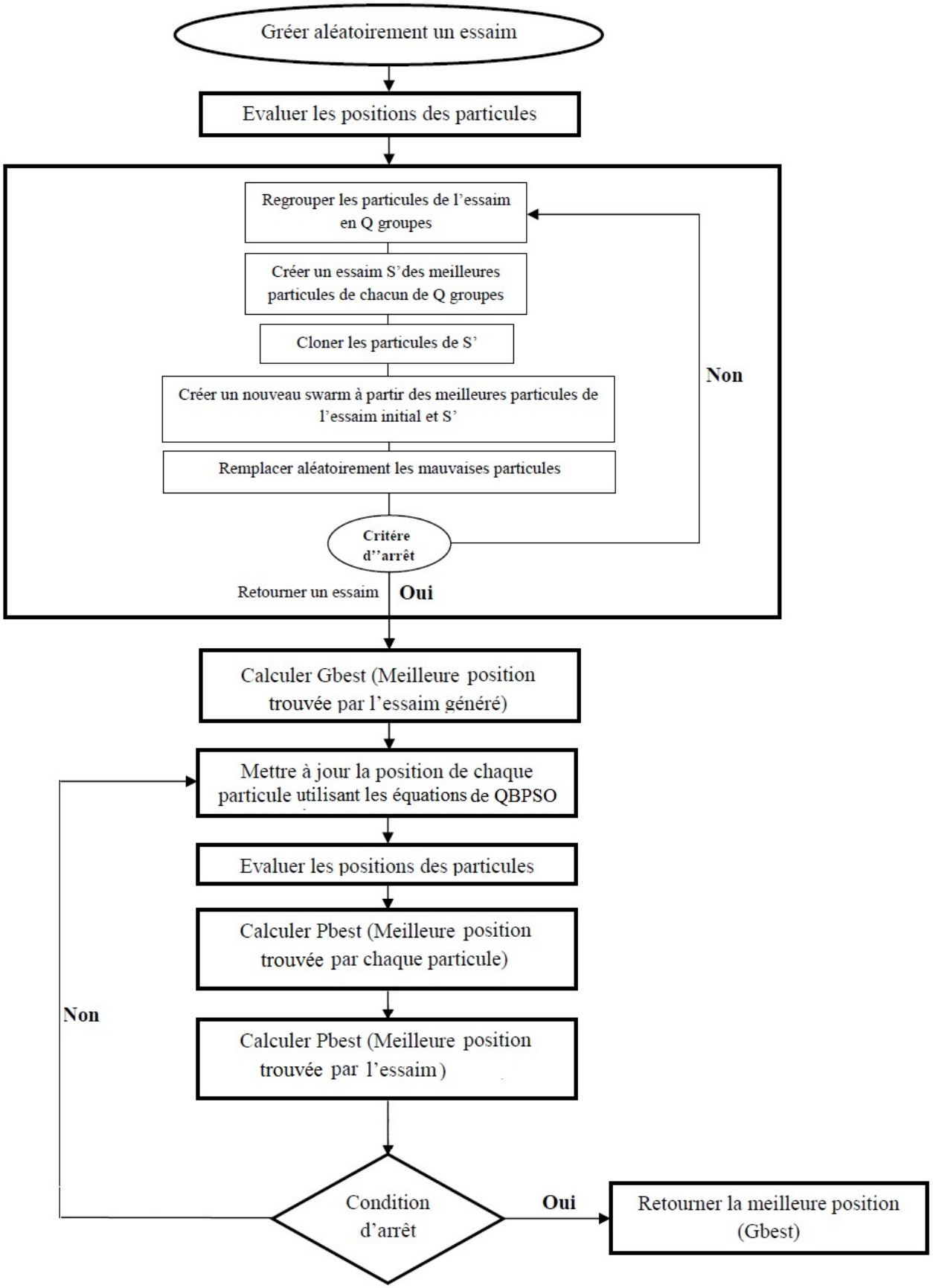

FIGURE 3.2 – Schéma fonctionnel de l'algorithme de sélection d'attributs BC-QPSO.

L'idée d'hybridation est d'améliorer la performance de recherche et la convergence rapide vers la meilleure solution, il s'agit de profité des avantages des deux algorithmes l'algorithme de sélection clonale et l'algorithme BQPSO, en effet l'objectif de notre approche est d'améliorer la qualité du swarm par une nouvelle stratégie d'initialisation des particules, en insérant les opérations de l'algorithme de sélection clonale dans l'algorithme QBPSO, dans le but d'assurer une bonne couverture de l'espace de recherche par les particules.
En d'autre terme, à chaque exécution du QBPSO, au lieu de suivre l'algorithme sur un swarm créer aléatoirement, on créer un swarm amélioré, l'algorithme BC-QPSO s'exécute en étapes suivantes :

- **Étape 1 :** Génération de la population initiale :
  Les particules P sont généré aléatoirement pour créer une population initiale de l'essaim S dans l'espace recherche.
- **Étape 2 :** Évaluation des particules :
  Pour toutes les particules d'un essaim S, calculer la valeur de la fonction fitness par la précision du classificateur K-ppv (Taux de classification).
- **Étape 3 :** Améliorer la qualité de l'essaim S :
  L'amélioration de l'essaim réalisé par une génération de nouvelle population des particules suivant l'algorithme 9.

Algorithm 9 Génération de la population des particules

Entrée : Un swarm S1 avec la valeur de la fonction fitness de chaque particule ;
Sortie : Un swarm S2 ;
Tant que la condition d'arrêt n'est pas vérifier **faire**
1. Regrouper les particules de S1 en P groupes basés sur le critère de proximité (en fonction des valeurs de la fonction fitness) ;
2. Choisir les meilleures particules dans chacun des P groupes et créer un sous swarm SQ ;
3. Cloner les particules de SQ ;
4. Créer la nouvelle génération du swarm S2 à partir des meilleures particules de SQ et S1 ;
5. Remplacer les mauvaises particules du nouveau swarm S2 de façon aléatoire ;
6. Remplacer S1 par S2 ;

Fin Tant que

Les détails de l'Algorithme 9 décrits comme suit :

1. **Regroupent des particules en P groupes :**
   - Le nombre de groupes est P = (Nombre de particule / 10).
   - Chaque groupe est représenté par un intervalle de réels.
   - La longueur de l'intervalle L = (Max_Fitness – Min_Fitness) / P :
   Les intervalles des groupes sont les suivants :
   Groupe 1 : [Min, Min+L [,
   Groupe 2 : [Min+L, Min+2L [,
   ... ,
   Groupe P : [Min+PL, Max]
   Affectation des particules aux différents groupes selon leur valeur de fitness.
   Exemple :
   Soit les un swarm de 20 particules avec leurs valeurs de fitness : (P1, 30), (P2, 32), (P3, 45), (P4, 10), (P5, 12), (P6, 5), (P7, 32), (P8, 62), (P9, 41) , (P10, 14), (P11, 45), (P12, 22), (P13, 9), (P14, 29), (P15, 58), (P16, 57), (P17, 26), (P18, 2), (P19, 51) , (P20, 16).

Le nombre de groupes est P = 20/10 = 2.
La longueur de l'intervalle L = (62-2)/2 =30.
- Groupe 1 : [2, 32[, contient les particules : P1, P4, P5, P6, P10, P12, P13, P14, P17, P18, P20
- Groupe 2 : [32, 62], contient les particules : P2, P3, P7, P8, P9, P11, P15, P16, P19

2. **Choisir les meilleures particules dans chacun des Q groupes :**
Choisir la meilleure particule de chacun des P clusters (groupes) et créer un swarm appelé SQ.
SQ = (P1, 30), (P8, 62)

3. **Cloner les particules de SQ :**
Y'a plusieurs façon de cloner des particules d'un swarm, on a proposé un mécanisme basé sur le principe suivant :
Pour toute particule P_i de SQ appliquer la procédure **Cloner** :

Cloner(P_i)
Générer un nombre aléatoire j $\in [1, d]$, d : *dimensions de vecteur position de* P_i
Si X_i[j]=1 **Alors** remise à 0
Sinon remise à 1 ;
Finsi
Retourner P_i

4. **Générer nouvelle génération (le Swarm S2) :**
Créer un nouveau swarm par la sélection des meilleures particules (qui ont le max fitness) à partir du S1 et SQ.
5. **Remplacer :**
- Choisir des mauvaises particules du swarm S2, qui ont des mauvaises valeurs de la fonction fitness (peut prendre 20%).
- Remplacer les mauvaises particules par des nouvelles particules générées aléatoirement.

- **Étape 4 :** Mettre à jour la position de chaque particule selon les équations suivantes :

- Déterminer le vecteur mbest :
mbest : est le vecteur des moyennes des meilleures position de touts les particules, le $j^{éme}$ bit de mbest est déterminé par les états des $j^{éme}$ bits de tous les Pbest des particules, le mbest est déterminé par la procédure **Get_mbest** suivante :

Get_mbest(Pbest)
Pour j=1 to l (l : taille de Pbest) **Faire**
sum=0 ;
Pour tout particule i **Faire**
sum=sum+Pbest[i][j] ;
Finpour
avg=sum/M ;
Si avg $\gg$ 0.5 **Alors** mbest[j]=1 ;**Finsi**
Si avg $\ll$ 0.5 **Alors** mbest[j]=0 ;**Finsi**
Si avg=0.5 **Alors**
Si rand( ) $\ll$ 0.5 **Alors** mbest[j]=0 ;
Sinon mbest[j]=1 ; **Finsi Finsi**
Finpour
Return mbest ;

- Déterminer le vecteur P_i :
P_i est le vecteur « position stochastique » généré par une opération de croisement entre $Pbest_i$ et Gbest par la procédure suivante : **Get_P** :

Get_P($Pbest_i$, Gbest)
Appliquer une opération de croisement
entre $Pbest_i$, Gbest
pour générer deux vecteur binaire $z_1 et z_2$;
Si rand( )$\ll$0.5 **Alors** $P_i = z_1$
Sinon $P_i = z_2$;
Finsi
Return P_i

- Calculer la probabilité de mutation Pr :
Nous pouvons obtenir la nouvelle position de la particule i par la transformation dans laquelle chaque bit dans le vecteur X_i de la particule i est muté avec la probabilité de mutation calculée par :

$$Pr = \begin{cases} 1 & \text{si } b/l \gg 1 \quad ,l; taille\ de\ vecteur\ position \\ b/l & \text{sinon} \end{cases} \tag{3.1}$$

$$b = d_H(X_i, P_i) = \beta(d_H(X_i, mbest))ln(1/u) \tag{3.2}$$
$$\beta, u : \ Nombres\ aléatoires$$

La fonction $d_H(X_i, mbest)$ permette d'obtenir la distance de Hamming entre X_i et mbest. La distance Hamming est le nombre de bits différents dans les deux vecteurs.
β s'appelle le coefficient Contraction-Expansion, qui peut être réglé pour contrôler la vitesse de convergence.
u est nombre aléatoire entre 0 et 1.

- La mise à jour de X_i : obtenu par une mutation de P_i dans comme montre la procédure **Transf** avec la probabilité Pr, comme suit :

Transf(P_i,Pr)
Pour touts bit de P_i **faire**
Si rand()$\ll$Pr **Alors**
Si etat de bit est 1 alors remise à 0 ;
Sinon remise à 1 ;
Finsi
Finsi
Finpour
X_i=P_i
Return X_i

- Evaluer toutes les particules de l'essaim.

- **Etape 5 :** Calculer Pbest et Gbest :

$$Pbesti(t+1) = \begin{cases} xi(t+1) & \text{si } f(xi(t+1)) \geq f(Pbesti(t)) \\ Pbesti(t) & \text{sinon} \end{cases} \tag{3.3}$$

$$Gbest(t+1) = argmax_{Pbest_i} f(Pbest_i(t+1)), 1 \leq i \leq N \tag{3.4}$$

- **Étape 6 :** Répéter l'étape 4 à 5 jusqu'à ce que le critère d'arrêt être satisfait.

3.3 Expérimentation

3.3.1 Matériels et logiciels utilisés

Notre approche a été implémentée et exécutée en utilisant :

- Un ordinateur de 4GO de RAM et un processeur Intel Celeron 3.6MHz.
- Java sous Eclipse.
- Logiciel WEKA.

Weka

(Waikato Environment for Knowledge Analysis) est une suite populaire de logiciels d'apprentissage automatique, écrite en Java, développée à l'université de Waikato, Nouvelle-Zélande, WEKA est un Logiciel libre disponible sous la Licence publique générale GNU (GPL).
L'espace de travail Weka contient une collection d'outils de visualisation et d'algorithmes pour l'analyse des données et la modélisation prédictive, allié à une interface graphique pour un accès facile de ses fonctionnalités.
Il est utilisé dans beaucoup de domaines d'application, en particulier, le domaine médicale, l'éducation, l'économie, les sciences sociales, etc.

Avantages de Weka : Les principaux points forts de Weka sont qu'il :

- Est librement disponible sous la licence publique générale GNU.
- Est très portable car il est entièrement implémenté en Java et donc fonctionne sur quasiment toutes les plateformes modernes.
- Contient une collection de d'algorithme de prétraitements de données, de classification, de clustérisation, etc.

3.3.2 Jeux de données de tests

Nous avons utilisé des données provenant de l'UCI Machine Learning Repository, l'UCI Machine Learning Repository contient une collection de bases et de générateurs de données qui sont utilisés par la communauté de l'apprentissage automatique (Machine Learning) pour l'analyse empirique des algorithmes d'apprentissage automatique. L'archive a été créée en 1987 à l'Université de Californie à Irvine (UCI).[11]
Le détail des jeux de données choisis est présenté dans le Tableau 3.1, le choix de ces jeux de données est basé sur le nombre d'attributs et le nombre d'instances.

BDD	Nb instances	Nb attributs	Description
Semeon	1593	257	Chaque personne a écrit tous les chiffres [0-9] deux fois sur papier, puis scannés, étirés dans une boîte rectangulaire dans une échelle de gris de 256 valeurs (chiffre représenté par 256 attributs binaires).
MUSK	476	168	Un ensemble de 92 molécules dont 47 sont jugés par des experts humains comme des muscs et les 45 molécules restantes non musquées.
LIBRAS	360	91	L'ensemble de données contient 15 classes de 24 instances chacune. Chaque classe fait référence à un type de mouvement manuel dans LIBRAS.
Audiology	226	70	Dans le cadre de la norme de 18 semaines de recommandation sur le traitement des maladies audiologies, l'ensemble de données contient des informations sur les différentes «étapes» du voyage pour les patients d'audiologie.

Tableau 3.1 – Jeux de données pour le teste

3.3.3 Critères de comparaison

Nous avons choisi pour comparaison la performance de notre approche par rapport aux autres présentées dans la littérature Les critères suivants :

- Moyen nombre d'attributs.
- Moyen taux de classification.
- Max taux de classification.

3.3.4 Réglage des paramètres

Le choix des paramètres jouer un rôle primordial dans l'efficacité et la performance de l'algorithme, il n'y a pas des valeurs standards des paramètres des algorithmes de sélection, ces valeurs dépendent des études empiriques et ses résultats.
Dans cette approche on s'intéresse à la taille de population, le nombre d'itérations et le nombre d'exécutions qui représente le nombre de fois que l'algorithme est exécuté.
Nous présentons ci-dessous les différents paramètres appliqués aux jeux de données :

- Critère d'arrêt (nombre d'itérations) : 10, 20, 30, 50, 100.
- Nombre de particules : 10, 20, 40, 50.
- Nombre d'exécution fixé par : 30 exécutions afin de prendre la moyenne de 30 exécutions.

3.3.5 Résultats expérimentaux

Dans cette section afin de valider notre approche BC-QPSO, nous avons refait les même testes sur les approche BPSO, BQPSO, en fixant les même paramètres.

3.3.5.1 Étude comparatif

Pour notre approche CB-QPSO nous avons observé des bons résultats avec les paramètres :
- Nombre d'itérations : 20.
- Nombre de particules (taille de population) : 20.
Le tableau 3.2 montre les résultats de tests selon les paramètres choisis :

Jeux de données	NTA	TCTA	Approche	NMA	MTC	TMC
Semeon	257	86.79	**BPSO**	128.2	92.66	92.75
			BQPSO	130.1	92.54	91.82
			BC-QPSO	129.6	92.66	91.45
MUSK	168	92.22	**BPSO**	80.26	99.15	98.66
			BQPSO	82.83	98.94	97.67
			BC-QPSO	82.56	99.15	98.27
LIBRAS	91	85.83	**BPSO**	42	89.16	88.60
			QPSO	44.03	88.88	87.32
			BC-QPSO	44.16	89.44	88.09
Audiology	70	77.87	**BPSO**	36.7	77.43	75.94
			QPSO	43.86	77.87	73.11
			BC-QPSO	35.73	76.99	74.54

Tableau 3.2 – Comparaison de résultats des approches implémentés.

NTA : Nombre Total d'Attribut.
TCTA : Taux de Classification utilisant Tout les Attributs de la base de test.
TMC : Taux Moyenne de Classification.
MTC : Meilleur taux de classification.
NMA : Nombre Moyen d'Attributs sélectionnés.

Discussion des résultats du tableau :
Dans le tableau 3.2 la comparaison est basée sur le nombre réduit des attributs par rapport à l'ensemble d'attributs d'origine, et le taux maximum de classification réalisé par le modèle d'apprentissage en préservant les attributs pertinents comparée au taux de classification utilisant touts les attributs , ainsi que la moyenne des max taux de classification atteints dans 30 exécutions.
Notre approche offre des bons résultats comparant avec les algorithmes BPSO et BQPSO surtout pour des bases de données de grande taille :

- BC-QPSO a réussit à réduire la taille de l'ensemble d'attributs par trouver le petit sous-ensemble optimal, avec une convergence plus rapide vers le taux de classification maximum.
- On remarque aussi que BPSO est compétitif à BC-QPSO en termes de taille de solution trouvée ainsi que le taux de classification maximum.
- Pour QBPSO malgré ça convergence rapide vers le sous-ensemble optimal avec la base de données « Audiology », mais reste en générale un peu moins performant pour les bases de données de grande taille par rapport aux BPSO et BC-QPSO.

3.3.5.2 Étude de l'effet du nombre de particules sur le taux de classification maximum

Les figures suivantes présentent les résultats expérimentaux obtenus en appliquant les trois approches sur les quatre jeux de données présentées en avant, en se basant sur le max du taux de classification évalué par rapport au nombre de particules en fixant à chaque fois de nombre d'itérations de la boucle de l'algorithme :

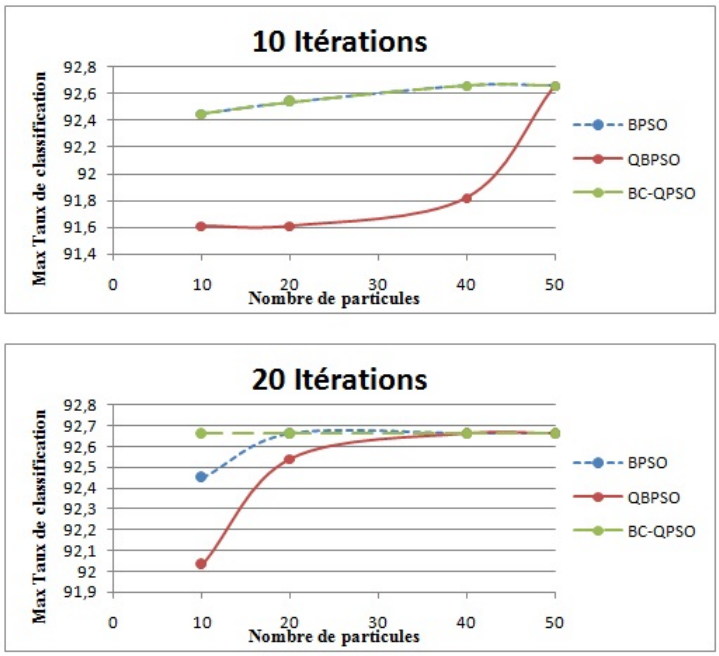

FIGURE 3.3 – Évolution de taux de classification maximum par rapport au nombre de particules pour la base Semeon.

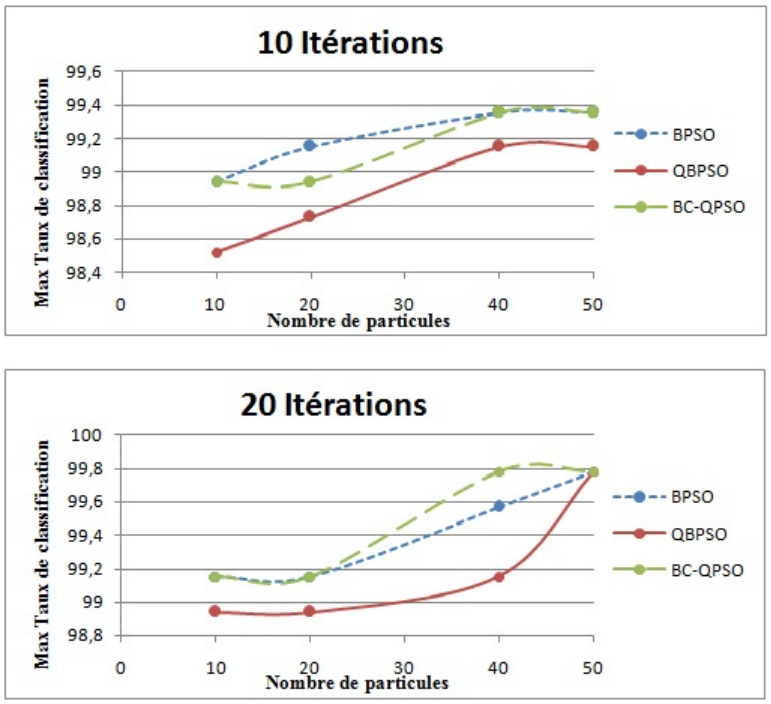

FIGURE 3.4 – Évolution de taux de classification maximum par rapport au nombre de particules pour la base MUSK.

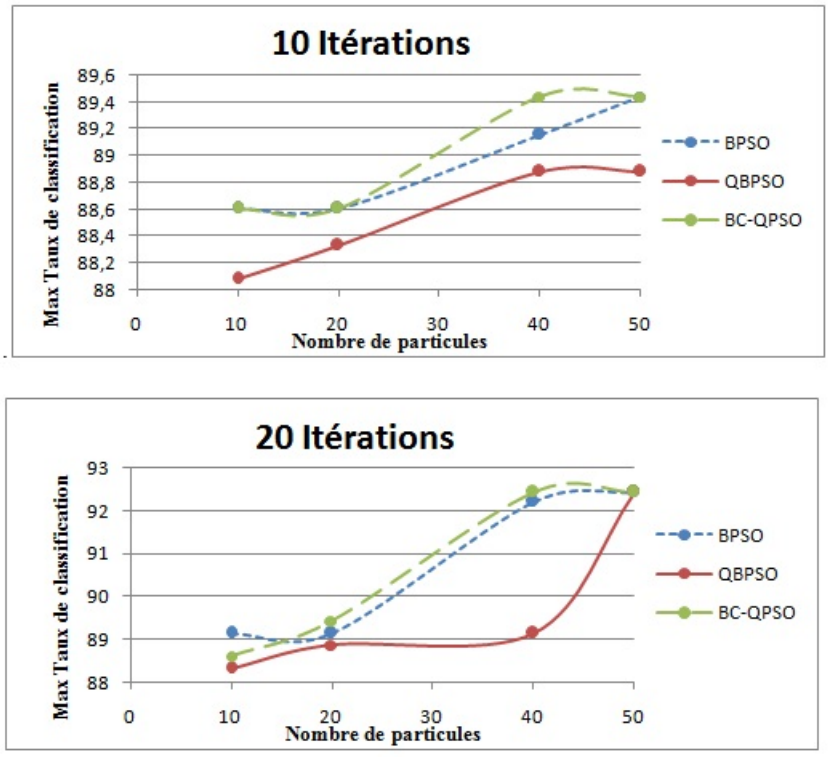

FIGURE 3.5 – Évolution de taux de classification maximum par rapport au nombre de particules pour la base LIBRAS.

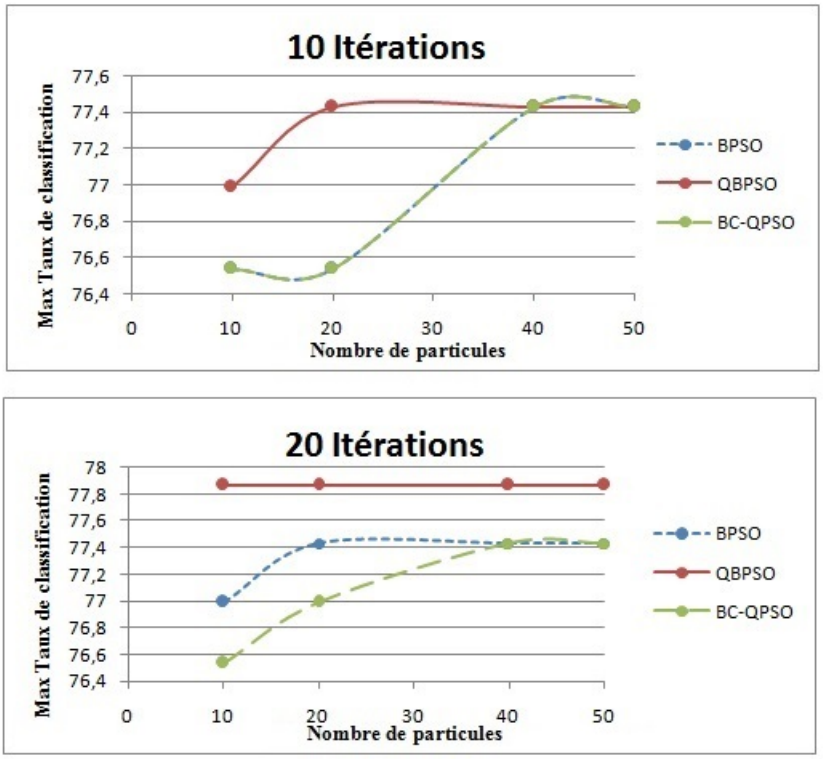

FIGURE 3.6 – Évolution de taux de classification maximum par rapport au nombre de particules pour la base Audiology.

Discussion des résultats :
A partir des courbes des figures qui montre l'évolution du taux de classification par rapport au nombre de particules utilisé (taille de la population) on remarque que BC-BQPSO donne des bonnes résultats utilisant une population de petite taille compétitive à BPSO ,contrairement à BQPSO, qui a besoin d'une population assez grand pour mieux chercher la solution optimal, dans la majorité des bases testées.
Dans le cas de la base de données « Audiology » l'algorithme BQPSO montre sont efficacité avec une population très réduite par rapport aux algorithmes BPSO, BC-QPSO.

3.3.5.3 Étude de convergence de l'approche BC-QPSO

Les figures suivantes présentent les résultats expérimentaux obtenus en appliquant les trois approches sur les quatre jeux de données, en se basant sur le taux maximum de classification évalué par rapport au nombre d'itérations en fixant cette fois le nombre de particules :

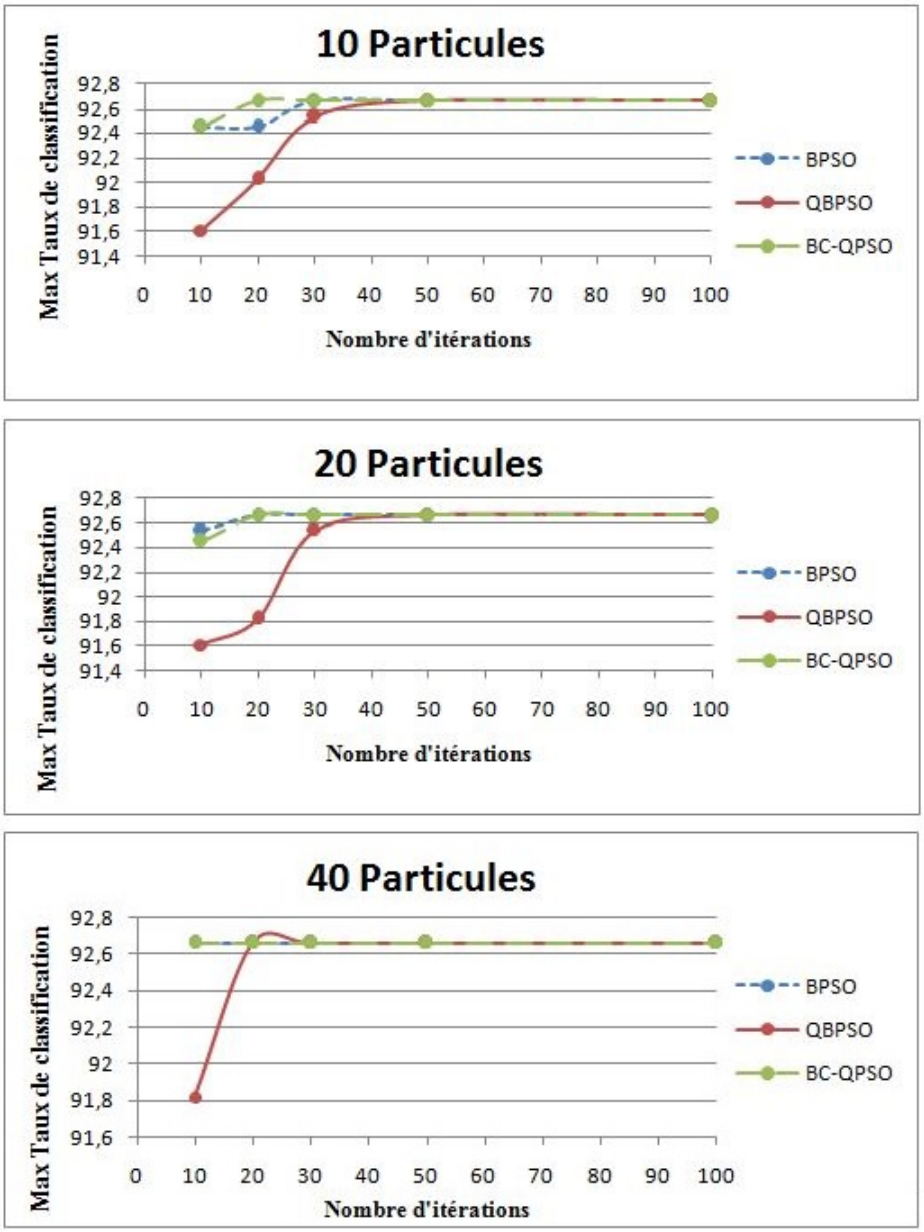

FIGURE 3.7 – Évolution de taux maximum de classification par rapport au nombre d'itérations pour la base Semeon.

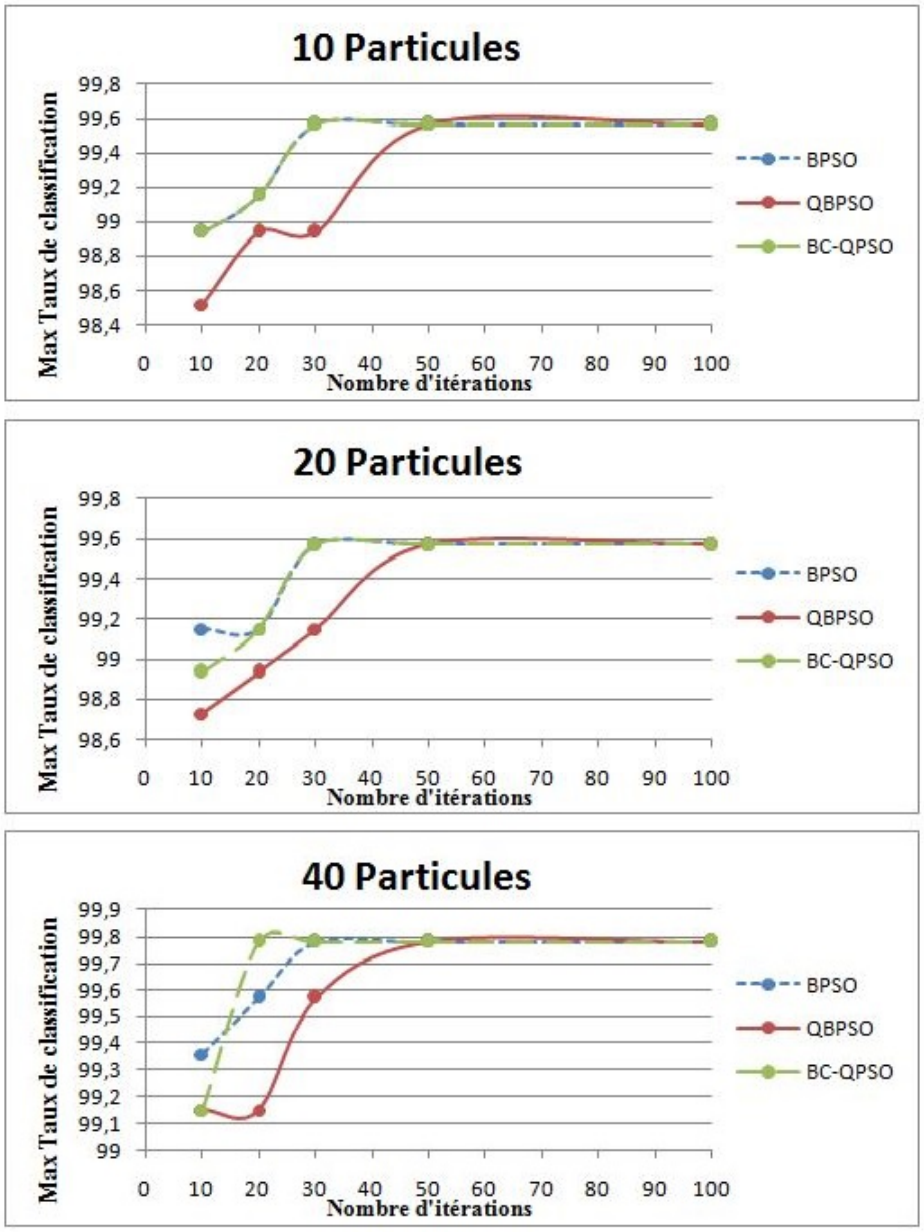

FIGURE 3.8 – Évolution de taux maximum de classification par rapport au nombre d'itérations pour la base MUSK.

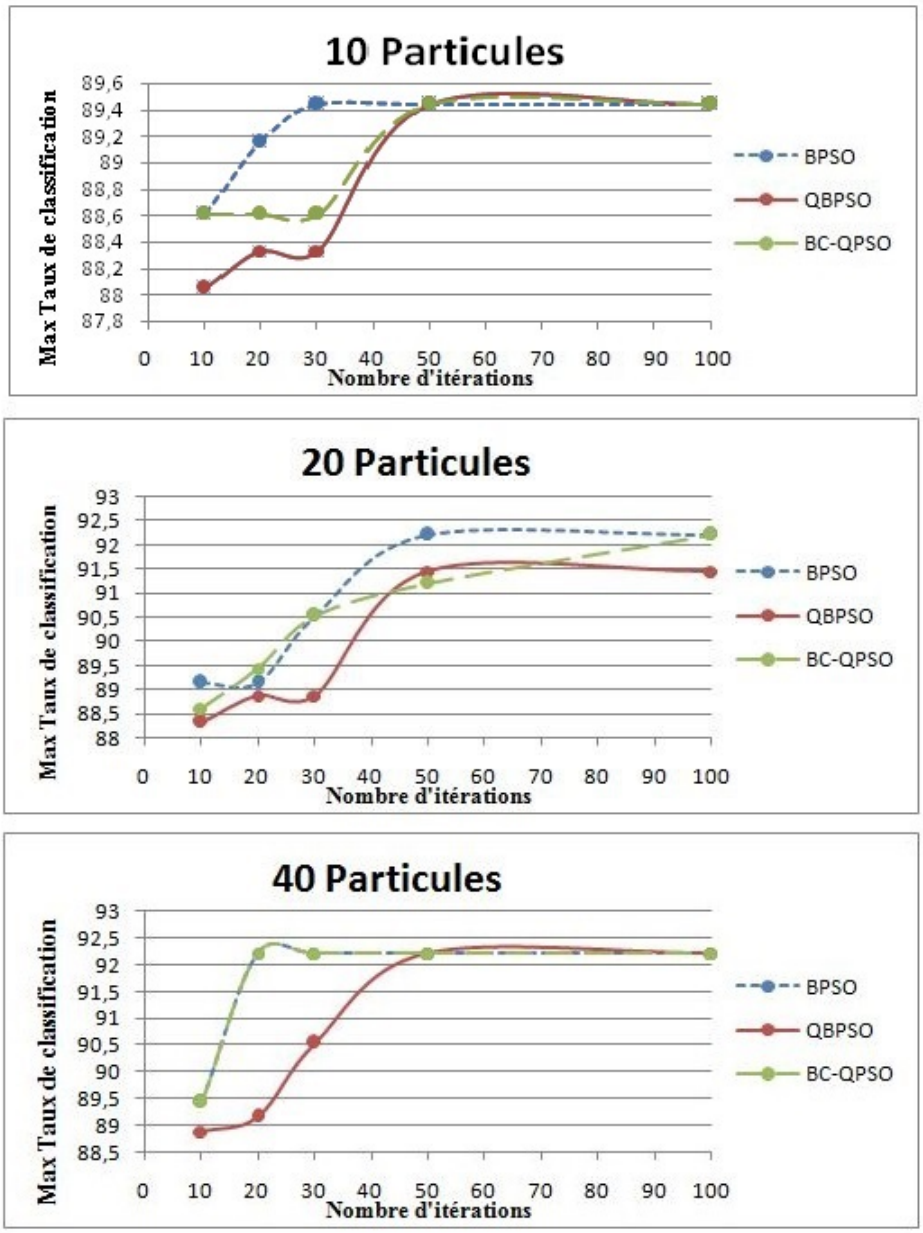

FIGURE 3.9 – Évolution de taux maximum de classification par rapport au nombre d'itérations pour la base LIBRAS.

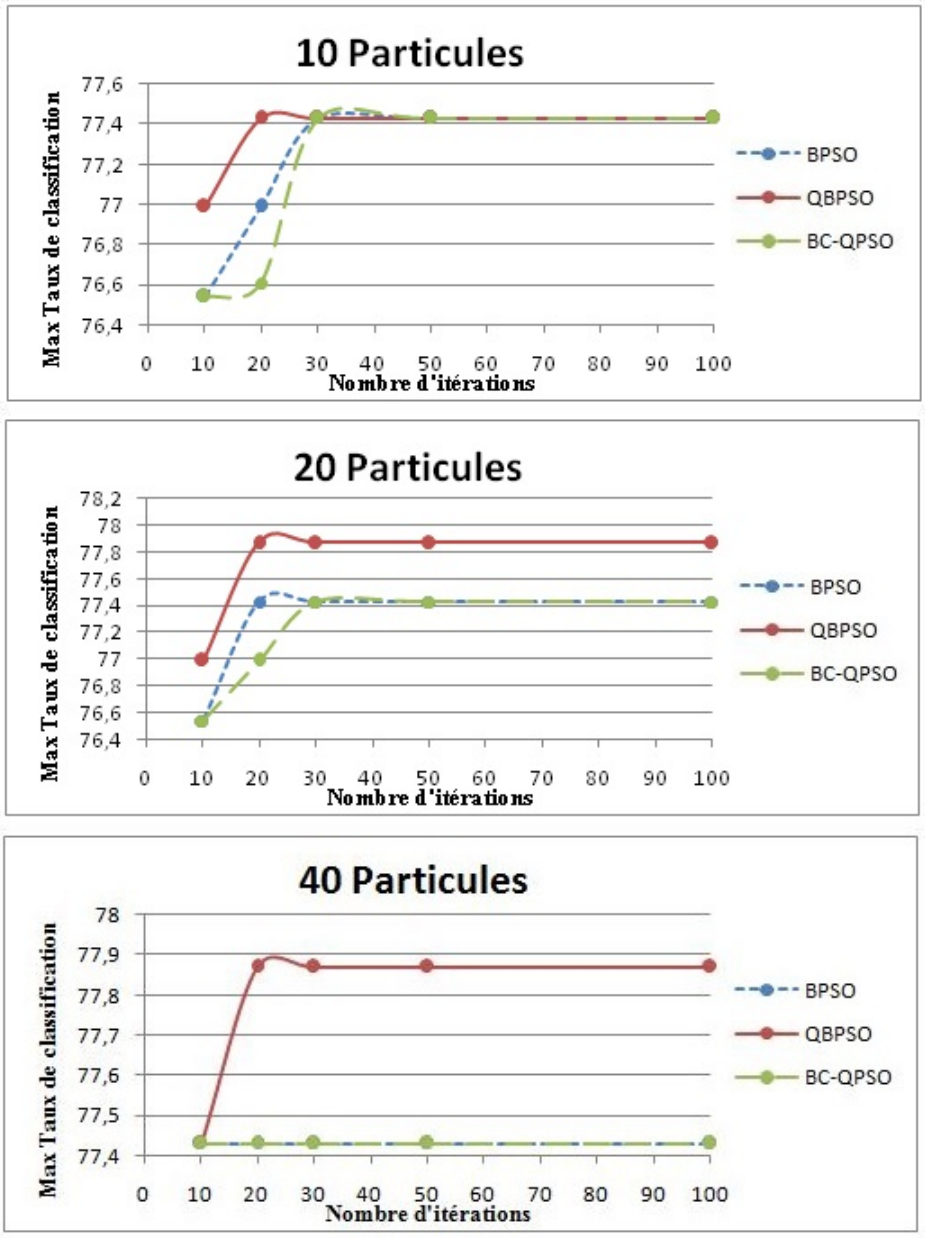

FIGURE 3.10 – Évolution de taux maximum de classification par rapport au nombre d'itérations pour la base Audiology.

Discussion des résultats :

Dans les courbes reportées dans les figures on a représenté l'évolution de taux de classification en fonction du nombre d'iération des trois algorithmes comparés dans notre étude, on a remarqué que le taux de classification atteint une valeur maximale par BC-QPSO et BPSO plus rapidement (Convergence rapide), par contre QBPSO n'obtient pas la meilleure solution que dans l'itération 30 ou après surtout pour les grands base de données, par contre pour les bases les de petite taille comme « Audiology » QBPSO marque son efficacité par rapport aux BPSO et BC-QPSO.

3.4 Interface graphique

L'interface graphique est une surcouche permettant d'utiliser le code sans programmer directement, son but est de simplifier la tâche de l'utilisateur en lui donnant accès directement aux différents paramètres et en lui apportant une visualisation graphique des résultats et une comparaison avec des courbe d'évalutation de taux de classification en fonction de nombre d'itération.
Elle est programmée par Java.

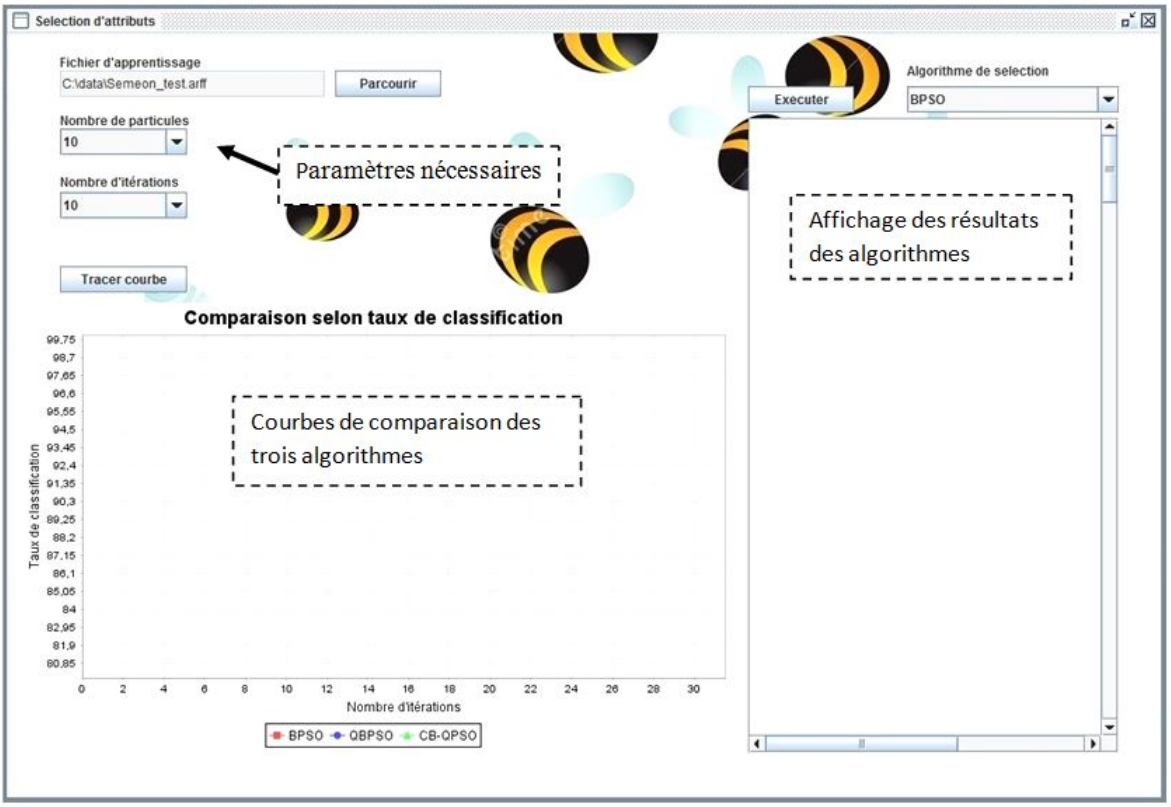

FIGURE 3.11 – Interface graphique de l'application "Sélection d'attributs".

3.5 Conclusion

Dans ce chapitre nous avons proposé une nouvelle approche hybride BC-QPSO pour la sélection d'attributs dans la classification des données, ensuite la comparer avec d'autres métaheuristiques de la littérature telles que BPSO et BQPSO.
Les résultats de comparaison montrent l'efficacité de l'approche proposée BC-QPSO par rapports aux autres approches, elle trouve des bonnes résultats dans un temps réduit (elle converge vers la solution optimale rapidement), on remarqué une réduction importante de la taille de l'ensemble des attributs sélectionnés qui donne généralement des taux de classification meilleure que l'utilisation de l'ensemble totale des attributs des bases de tests.

Conclusion et perspectives

Plusieurs paramètres peuvent influer les performances de la tâche de classification supervisée de données, le nombre et la qualité des attributs extraits à partir des entités considérés peut être considères parmi les paramètres les plus importants.
La sélection d'attributs consiste à choisir parmi un ensemble de grande taille un sous-ensemble d'attributs intéressants ou pertinent pour le problème étudie, elle réduit le nombre des attributs, élimine les attributs non pertinents, redondants ou bruités.
En effet, la sélection d'attributs fait partie des problèmes d'optimisation NP-difficile qui peut être résolu par les métaheuristiques.
Dans notre travail nous avons proposé une nouvelle approche BC-QPSO pour la résolution du problème de sélection d'attributs basée sur la combinaison de deux métaheuristiques proposées dans la littérature BQPSO et le système immunitaire artificiel, le but de notre approche est d'améliorer la performance de la méthode BQPSO par l'intégration du mécanisme de sélection clonale afin de mieux explorer l'espace de recherche. Les expérimentations sont réalisées sur des bases de données de l'UCI (University of California, Irvine), cette approche hybride est comparés avec les approches BPSO et BQPSO proposées dans la littérature.

Les résultats de nos expérimentations montrent que notre approche BC-QPSO proposée est compétitive aux approches testées, elle présente une convergence rapide à la solution optimale, elle permet d'atteindre un taux de classification élevé dans un temps réduit utilisant un sous-ensemble réduit d'attributs.

De plus, la méthode de sélection BPSO donne généralement des résultats meilleurs que ceux de la méthode basée sur QBPSO en termes d'augmentation de taux de classification et de réduction du nombre d'attributs sélectionnés.

Comme perspectives :
Il est certain que la solution proposée n'est pas la meilleure d'une manière absolue, plusieurs améliorations sont envisageable et plusieurs point peuvent être étudiées dans le cadre de travaux future, tel que :
- Proposés d'autres hybridations basé sur BQPSO et d'autres métaheuristiques comme les algorithmes de colonie de fourmi et l'algorithme génétique.
- Concevoir et développer une plateforme de résolution de problèmes d'optimisation de différents complexités dans laquelle nous intégrons notre méthodes et d'autres méthodes existantes pour fournir un moyen permettant de facilité la mise en œuvre de différentes métaheuristiques pour la résolution des problèmes d'optimisation et en particulier la sélection d'attributs.

Annexes

Annexe A

La classe du classifieur Kpp-v **Classification**.

```
package BCQPSO;
import weka.core.*;
public class Classification {
    private  int nb_instances;
    private int nb_instances_correcte=0;
    private double taux_de_classification;

    Classification(Instances instances) {
        try {
        // ---------------------- Classifier-----------------------------
        IBk  KNN = new IBk();
        KNN.setKNN(5);
        KNN.buildClassifier(instances);
        // ---------------------- Results of classification----------------------------
        this.nb_instances= instances.numInstances();
        this.nb_instances_correcte=0;
        for (int i=0;i<nb_instances;i++){
        if (KNN.classifyInstance(instances.instance(i))== instances.instance(i).value(instances.numAttributes()-1))
        { this.nb_instances_correcte++; }}
        this.taux_de_classification=((double) nb_instances_correcte*100)/ (double) nb_instances;
        }
        catch(Exception ex){
        ex.printStackTrace();}
        }

    public int Get_nb_instances(){ return nb_instances; }

    public int Get_nb_instances_correcte(){ return nb_instances_correcte;}

    public double Get_taux_de_classification(){ return taux_de_classification; }
    }
```

Annexe B

La classe principale **Test**.

```
package BCQPSO;
import java.io.FileReader;
public class Test {
public static void main(String[] args) {
        // --------------- Lecture du fichier d'apprentissage ---------------
        try {
        //-----------------Définition du fichier d'apprentissage -------------
            String File_Learning="c:/data/Semeon.arff";
            int Nb_iterations=20;
            int Nb_Particles=20;
            int Nb_executions=30;
            FileReader Exemple = new FileReader(File_Learning);
            Instances instances = new Instances(Exemple);
            System.out.println("Sélection des attributs du fichier ----: "+ File_Learning +" ----"+"\n");
            System.out.println("Le nombre total d'attributs sans attribut classe: "+(instances.numAttributes()-1));
            System.out.println("Le nombre total de particules: "+Nb_Particles);
            System.out.println("L'algorithme BC-QPSO ");
    double Moy_nb_attributs_Gbest=0.0;
    double Moy_Fitness_Value_Gbest=0.0;
    double Max_Fitness_Value_Gbest=0.0;
    for(int i=1;i<=Nb_executions;i++){
        System.out.print("execution: "+i+":"+"\n");

        //--------------------Génération de la population initiale--------------------------------
        Swarm S0 = new Swarm(Nb_Particles,Nb_iterations,File_Learning);
        for(int j=1;j<=Nb_iterations;j++){
        Swarm S1 = new Swarm(Nb_Particles,Nb_iterations,File_Learning);
        S1=S0.Creation_Swarm1();
        S1.cloner1();
        S1=S0.Nouvelle_Generation(S1);
        S0=S1.remplacer_particules(Nb_Particles);
        S1=null;}

        //--------------------Lancer l'algorithme QBPSO----------------------------------------
        S0.Recherche();
        System.out.println("Gbest"+S0.Gbest);
        System.out.println("Valeur_Fitness_Gbest"+S0.Fitness_Value_Gbest);
        System.out.println("Nb_Attributs_Gbest"+S0.calcul_nb_attributs_Gbest());
        Moy_nb_attributs_Gbest+=S0.nb_attributs_Gbest;
        Moy_Fitness_Value_Gbest+=S0.Fitness_Value_Gbest;
        if(S0.Fitness_Value_Gbest>Max_Fitness_Value_Gbest){
        Max_Fitness_Value_Gbest = S0.Fitness_Value_Gbest;}
        S0=null;
        System.gc();}

    System.out.println("------- Résultat BC-QPSO ------------------------"+"\n");
    System.out.println("Moyen_Nb_attributs_Gbest = "+(Moy_nb_attributs_Gbest/Nb_executions));
    System.out.println("Moy_Taux_Classification= "+ (Moy_Fitness_Value_Gbest/Nb_executions));
    System.out.println("Max_Taux_Classification"+ Max_Fitness_Value_Gbest+"\n");
    System.out.print("\n");}
            catch(Exception ex){
            ex.printStackTrace();}}}
```

Annexe C

La classe **Swarm** contient toutes les méthodes utilisées.

```
package BCQPSO;
import java.util.Vector;
import java.util.Random;
import weka.core.Instances;
import java.io.*;

    public class Swarm {

        // ---------- Particle class -------------------------------
    public class Particle {

        // ----------Attributes of Particle ------------------------
        private Vector<Integer>  Position;
        private  Vector<Integer> Pbest;
        private double Fitness_Value_Pbest=-1;
        private int nb_attributs_Pbest=0;
        private Instances instances_locales;
        private String File_Learning;
        @SuppressWarnings("unused")
        private double Fitness_Value_Position=-1 ;

        // ---------- Methods of Particle --------------------------
        Particle(Vector<Integer> Position,String File_Learning ){

        // ---------- Initialization -------------------------------
            this.Position = Position;
            this.File_Learning = File_Learning;
            this.Pbest=Position;
            this.Classification( File_Learning);}

       // ---------- Classification method by vector Position ------
        public void Classification(String File_Learning){

       // ----------Preparing the local instance--------------
       // ----------Local representation of the learning file-------
        try {
            FileReader Exemple_locale = new FileReader(this.File_Learning);
            this.instances_locales = new Instances(Exemple_locale );

      // ----------Delete unselected attributes (which corresponds to the zeros in the Position vector)--
            int m=0;
            for(int j=0;j< this.Position.size(); j++){
                if (this.Position.elementAt(j)==0){
                    instances_locales.deleteAttributeAt(m);}
                else{ m=m+1; }}

     // ----------Defining the Last Attribute as a Class Attribute -------------
            instances_locales.setClassIndex(instances_locales.numAttributes() -1);

     // ----------Apply classifier to calculate Fitness_value_Pbest
            Classification A = new Classification(instances_locales);

    // ----------Update "Pbest" and "Fitness_Value_Pbest"-----------
            Double Taux_classification = A.Get_taux_de_classification();
            this.Fitness_Value_Position = Taux_classification;
            A=null;
            System.gc();
            if (this.Fitness_Value_Pbest < Taux_classification){
                this.Pbest=this.Position;
                this.Fitness_Value_Pbest = Taux_classification;
                this.nb_attributs_Pbest = this.calcul_nb_attributs__du_Vecteur(this.Pbest);}
            else{ if (this.Fitness_Value_Pbest == Taux_classification){
                    if(this.nb_attributs_Pbest > this.calcul_nb_attributs__du_Vecteur(this.Position)){
                        this.Pbest=this.Position;
                        this.Fitness_Value_Pbest = Taux_classification;
                        this.nb_attributs_Pbest=this.calcul_nb_attributs__du_Vecteur(this.Position);}}}
                        Exemple_locale.close();}
        catch(Exception ex){
        ex.printStackTrace();}}

    // ----------Method for calculating Pi-------------
        public Vector<Integer> Pi(Vector<Integer> p1,Vector<Integer> p2){
```

```
    public Vector<Integer> Pi(Vector<Integer> p1,Vector<Integer> p2){
            Vector<Integer> p_1=new Vector<Integer>();
            Vector<Integer> p_2=new Vector<Integer>();

// ----------Crossing operation between p1 and p2-------------
        int pos = Swarm.rand.nextInt(Nb_Attributes);
        double kk2 = Swarm.rand.nextDouble();
        for(int i=0 ; i<=pos;i++){
            p_1.addElement(p1.elementAt(i));
            p_2.addElement(p2.elementAt(i));}
            for(int i=(pos+1);i<Nb_Attributes;i++){
                p_1.addElement(p2.elementAt(i));
                p_2.addElement(p1.elementAt(i));}
            if (kk2<0.5){return p_1;}
            else {return p_2;}}

//-------------------------------------------------------------------------------------
    public int Distance_Hamming(Vector<Integer>v1,Vector<Integer>v2){
        int d=0;
        for(int i=0;i<v1.size()-1;i++){
            if(v1.elementAt(i)!=v2.elementAt(i)){d++;}}
    return d;}

//----------Updating the position of the particle - used for learning (Apprentissage)-------
    public void update_Position(String File_Learning){
        Vector<Integer> Nouvelle_Position =new Vector<Integer>();
        this.File_Learning = File_Learning;
        int b;
        int u1=Swarm.rand.nextInt();
        b=(int) (Distance_Hamming(this.Position,Get_mbest())*(Math.log(1/u1)/Math.log(2)));
        double pr=0;
        if ((b/Nb_Attributes)>1){pr=1;} else{pr=(b/Nb_Attributes);}
        int randnb = Swarm.rand.nextInt();
        for(int i=0;i<Nb_Attributes-1;i++){
            if(randnb<pr){
            if((this.Pi(Pbest,Gbest).elementAt(i))==0){Nouvelle_Position.addElement(1);}
             else{Nouvelle_Position.addElement(1);}}
             else{Nouvelle_Position.addElement(this.Position.elementAt(i));}}
         Nouvelle_Position.addElement(1);
         this.Position = Nouvelle_Position;
         this.Classification(File_Learning);}

//-------------------------------------------------------------------------------------
    public void afficher_Postions(){System.out.println(Position);}

//----------Calculate the number of attributes of a vector -----
    public int calcul_nb_attributs__du_Vecteur(Vector<Integer> V){
        int nb_attributs=0;
        for(int k=0;k<V.size()-1; k++){
            if (V.elementAt(k)== 1){
                nb_attributs++;}}
    return nb_attributs;}}

//----------End of class Particle -------------------

//----------Swarm class-----------------
//----------Attributes of Swarm---------------
public Vector <Particle> Vp=new Vector<Particle>();     //---------- Vector of particles
public Vector<Integer> Gbest; //
public double Fitness_Value_Gbest=-1;                   //-----------Classification rate using the Gbest vector
public int nb_attributs_Gbest=0;
public int Nb_Particles=0;
public int Nb_Attributes=0;                            //------------Number of attributes of the learning file Example
private FileReader Exemple;                            //------------Learning file
private Instances instances;                           //------------Instance of the Learning File
private String  File_Learning="";
public int Nb_iteration=0;
static Random rand = new Random( 130L );
```

```
//----------Methods of Swarm

//----------1st constructor-------------------------------------------------------------------

    public Swarm(int Nb_Particles, int Nb_iterations, String File_Learning){
    try {
        this.File_Learning=File_Learning;
        this.Exemple = new FileReader(File_Learning);
        this.instances = new Instances(this.Exemple);}
    catch(Exception ex){
    ex.printStackTrace();}

    this.Nb_Particles = Nb_Particles;
    this.Nb_Attributes = instances.numAttributes();
    this.Nb_iteration = Nb_iterations;
    this.Particles_Initialisation();
    this.Update_Gbest();}

//----------2nd  constructor-----------------------------------------------------------
    public Swarm(Vector<Particle> vpart, int nb_iteration, String file_Learning) {
    this.Vp=vpart;
    this.Nb_iteration=nb_iteration;
    this.File_Learning=file_Learning;}

    public void Particles_Initialisation_pertinent(){

//----------Creating the Particle Vector------------
        Vp=new Vector<Particle>();

//----------Creating Particles--------------
        for(int i= 0; i< this.Nb_Particles;i++){

//----------Position vector-----------
            Vector<Integer> P = new Vector<Integer>();
            for(int j=0;j< this.Nb_Attributes-1; j++){

//---------- Initialization of particles
                if(j==i){P.addElement(0);}
                else P.addElement(1);}
//----------Leaving the last attribute of classes always 1
        P.addElement(1);

//----------Add particle to vector Vp---------
        Vp.addElement(new Particle(P,this.File_Learning));}}

    public void Particles_Initialisation(){

        Vp=new Vector<Particle>();

//----------Creating particles--------------
        for(int i= 0; i< this.Nb_Particles;i++){

        Vector<Integer> P = new Vector<Integer>();
            for(int j=0;j< this.Nb_Attributes-1; j++){
            if (Swarm.rand.nextInt(10) <5)
            {P.addElement(0);}
            else    P.addElement(1);}

        P.addElement(1);

        Vp.addElement(new Particle(P,this.File_Learning));}}

//----------Calculating the mbest vector-------------
    public Vector<Integer> Get_mbest(){
        Vector<Integer> mb=new Vector<Integer>();
        double r=Swarm.rand.nextDouble();
        for (int i =0;i<this.Nb_Attributes;i++){
            int som=0;
        for(int j=0;j<this.Nb_Particles;j++){
          som+= this.Vp.elementAt(j).Pbest.elementAt(i);}
          int avg= som /Nb_Particles;
          if (avg<0.5){mb.addElement(0);}
          if (avg>0.5){mb.addElement(1);}
          if (avg==0.5){
```

```
            if (r<0.5){mb.addElement(0);}
            else {mb.addElement(1);}}}
        return mb;}

    //----------Main method: Gbest subset search --------
        public void Recherche(){
            for(int n=0; n<this.Nb_iteration;n++){
                this.Update_Position(this.File_Learning);
                this.Update_Gbest();}}

    //----------Updating Particle Positions ----------------------------------------
        public void Update_Position(String File_Learning){
            for (int i =0;i<this.Nb_Particles;i++){
                Particle P = (Particle) Vp.elementAt(i);
                P.update_Position(File_Learning);}}

    //----------Updating Gbest ----------------------------------------------------
        public void Update_Gbest(){
            for (int i =0;i<this.Vp.size()-1;i++){
                Particle P = (Particle) Vp.elementAt(i);
                if (P.Fitness_Value_Pbest > this.Fitness_Value_Gbest){
                this.Gbest = P.Pbest;
                this.Fitness_Value_Gbest=P.Fitness_Value_Pbest;}
                else{
                    if (P.Fitness_Value_Pbest == this.Fitness_Value_Gbest){
                        if (P.nb_attributs_Pbest < this.nb_attributs_Gbest){
                            this.Gbest = P.Pbest;
                            this.Fitness_Value_Gbest=P.Fitness_Value_Pbest;}}}}
        this.nb_attributs_Gbest=this.calcul_nb_attributs_Gbest();}

    //----------Calculate the number of attributes in the Gbest vector ---------------
        public int calcul_nb_attributs_Gbest(){
            int nb_attributs=0;
            for(int k=0;k<this.Gbest.size()-1; k++){
                if (this.Gbest.elementAt(k)== 1){nb_attributs++;}}
        return nb_attributs;}
    //----------Show the particles of the swarm (vectors of the positions of each particle)----------------------------------------
        public void afficher_Particles(){
            System.out.print("Le nombre de Particules : "+this.Nb_Particles+"\n");
            System.out.print("Le nombre total d'attributs   : "+(this.Nb_Attributes-1)+"\n");
            for(int j=0;j< this.Nb_Particles; j++){
                System.out.print("\n");
                System.out.print("Particule " + j+" : ");
                Vp.elementAt(j).afficher_Postions();
                System.out.println ("Nb Attributs Pbest: "+Vp.elementAt(j).calcul_nb_attributs__du_Vecteur(Vp.elementAt(j).Pbest));
                System.out.println("Taux de classification : "+Vp.elementAt(j).Fitness_Value_Pbest +"\n");}}

    //----------Methods of Initial Population Generation (clonal mechanism)----------------------------------------------------------

    //----------Grouping clustered swarm particles--------------
        public Swarm Creation_Swarm1(){
            Double  min=this.Vp.elementAt(0).Fitness_Value_Pbest;
            Double max=this.Vp.elementAt(0).Fitness_Value_Pbest;
            Vector<Particle> Vpart=new Vector<Particle>();

    //----------Look for a min and max of the total fitness interval-----------
            for (int i=0;i<this.Vp.size();i++){
                if (this.Vp.elementAt(i).Fitness_Value_Pbest>=max){
                    max=this.Vp.elementAt(i).Fitness_Value_Pbest;   }
                if (this.Vp.elementAt(i).Fitness_Value_Pbest<=min){
                    min=this.Vp.elementAt(i).Fitness_Value_Pbest;}}

    //----------Create the swarm s1 from the best particles of each interval----------
            int nb_cluster=this.Vp.size()/10;
            double lang_interval =(max-min)/nb_cluster;
               while (min<max){

    //----------vs particle vector belongs to the cluster------------
                Vector<Particle> vs=new Vector<Particle> ();
                for(int i=0;i<this.Vp.size();i++){
                    if (this.Vp.elementAt(i).Fitness_Value_Pbest>=min && this.Vp.elementAt(i).Fitness_Value_Pbest<min+lang_interval){
```

```
                Vector<Integer> j=new Vector<Integer>();
                j=this.Vp.elementAt(i).Position;
                vs.addElement(new Particle(j,File_Learning));}}
                Vector<Integer> j=new Vector<Integer>();
             for (int i=0;i<Nb_Attributes; i++){j.addElement(1);}
             Particle max_vs=new Particle(j,this.File_Learning);
        for(int k=0;k<vs.size();k++){
        if (vs.elementAt(k).Fitness_Value_Pbest>=max_vs.Fitness_Value_Pbest){
          max_vs=vs.elementAt(k);}}

//----------Vpart Contains the max in each cluster-------------
            Vpart.add(new Particle(j,File_Learning));
            min+=lang_interval;}
            Swarm s11=new Swarm(Vpart,this.Nb_iteration,this.File_Learning);
            s11.Nb_Particles=Vpart.size();
    return s11 ;}

//----------Cloning and evaluating the particles of the swarm------------
    public void cloner1(){
        int j = Swarm.rand.nextInt(Nb_Attributes+1);
        for (int i=0;i<this.Nb_Particles;i++){
            Vector<Integer> nouvelle_pos= new Vector<Integer>();
            for(int k=0;k<this.Nb_Attributes-1;k++){
            if (k==j){
            if (this.Vp.elementAt(i).Position.elementAt(j)==1){
                nouvelle_pos.addElement(0);}
            else {nouvelle_pos.addElement(1);}}
            else{
            if (this.Vp.elementAt(i).Position.elementAt(j)==1){
                nouvelle_pos.addElement(1);}
            else {nouvelle_pos.addElement(0);}}}
            nouvelle_pos.addElement(1);
            this.Vp.elementAt(i).Position=nouvelle_pos;
            this.Vp.elementAt(i).Classification(File_Learning);}}

 //----------Creation of new generation from better Particles of two swarm-----------
     public Swarm Nouvelle_Generation(Swarm s){
         Swarm s1=null;
         s=new Swarm(Nb_Particles,Nb_iteration,File_Learning);
         Vector<Particle> vp=new Vector<Particle>();
         Particle max1=this.Vp.elementAt(0);
         Particle max2=s.Vp.elementAt(0);

 //----------Search max in 1st swarm s (which executes the method)---------
         for(int i=0;i<this.Vp.size();i++){
             if(this.Vp.elementAt(i).Fitness_Value_Pbest>max1.Fitness_Value_Pbest){
             max1=this.Vp.elementAt(i);}}

 //----------Search max in 2nd swarm (passed in parameter)-----------
         for(int i=0;i<s.Vp.size();i++){
             if(s.Vp.elementAt(i).Fitness_Value_Pbest>max2.Fitness_Value_Pbest){
             max2=s.Vp.elementAt(i);}}

 //----------Choose best in 1st swarm
         for(int i=0;i<this.Vp.size();i++){
             if(this.Vp.elementAt(i).Fitness_Value_Pbest==max1.Fitness_Value_Pbest){
             vp.addElement(new Particle(this.Vp.elementAt(i).Position,File_Learning));}}

 //----------Choose best in 2nd swarm------------
         for(int i=0;i<s.Vp.size();i++){
             if(s.Vp.elementAt(i).Fitness_Value_Pbest==max2.Fitness_Value_Pbest){
             vp.addElement(new Particle(s.Vp.elementAt(i).Position,File_Learning));}}
             int a=vp.size();
             s1=new Swarm(a,this.Nb_iteration,this.File_Learning);
             s1.Vp=vp;
     return s1;}
 //----------Replace the bad particles randomly----------
     public Swarm remplacer_particules(int nbr_particles){
         Vector<Particle> vp=new Vector<Particle>();
         for(int i=this.Vp.size();i<nbr_particles;i++){
             Vector<Integer> P = new Vector<Integer>();
             for(int j=0;j< this.Nb_Attributes-1; j++){
                 if (i==j){P.addElement(0);}
                 else {P.addElement(1);}}

 //----------Leaving the last attribute of classes always 1--------
         P.addElement(1);

 //----------Creation of the particle-----------
         vp.addElement(new Particle(P,this.File_Learning));}
             for(int k=0;k<vp.size();k++){this.Vp.addElement(vp.elementAt(k));}
         return this;}}
```

Bibliographie

[1] A.L.Blum et P.Langley. « Selection of relevant features and examples in machine learning ». In : *Artificial Intelligence* 97 (1997), p. 584–594.

[2] B.J.Clever. « Algorithms Nature-Inspired Programming Recipes ». In : *lulu Entrprises* (2011), p. 257–264.

[3] L.Nunes de Castro et F.J.Von Zuban. « Learning and optimization using the clonal selection principle ». In : *IEEE Transactions on Evolutionary Comptation* 6 (2002), p. 239–251.

[4] C.Blum et A.Roli. « Metaheuristics in Combinatorial Optimization : Overview and Conceptual Comparison ». In : *ACM an Computing Surveys* 35 (2003), p. 268–308.

[5] K.P. Syari et C.D Gelatt. « Optimization by simulated annealing ». In : *Journal of Science* 220 (1983), p. 671–680.

[6] C.Darwin. *The origin of apecies by means of natural selection.* New York : Mentor Reprint, 1959.

[7] C.F.Juang. « A hybrid of genetic algorithm and particle swarm optimization for recurrent network design ». In : *IEEE Trans Syst Man Cybern* 34 (2004), p. 997–1006.

[8] C.Guillaume. « Etudes des principaux algorithmes de data mining ». In : *Ecole ingnieurs en informatique EPITF* (2006).

[9] C.H.Papadimitriou. « The complexity of combinatorial optimization problems ». Doctorat. Ecole Polytechnique, 1976.

[10] C.H.Papadimitriou et K.Steiglitz. *Combinatorial Optimization - Algorithms and Complexity.* New York : Dover Publications, 1982.

[11] CL.Blake et CJ.Merz. « UCI repository of machine learning databases ». In : *CA : University of California ,Irvine, Department of Information and Computer Science* (1998), p. 203–232.

[12] C. Cotta et P. Moscato. « The k-feature set problem is W[2]-complete ». In : *Journal of computer ans system sciences* (2003), p. 686–690.

[13] D.Bell et H.Wang. « A formalism for relevance and its application in feature subset selection ». In : *Machine Learning* (2000), p. 175–195.

[14] D.E.Goldberg. *Genetic algorithms in search, optimization and Learning.* USA : Addison-Wesley, 1989.

[15] D.Goodman, L.Bogges et A.Watkins. « Artificial immune systeme classification of multiple class problems ». In : *Proceedings of Intelligent Engineering Systems* (2002), p. 179–184.

[16] DJ.Hand, H.Mannila et P.Smyth. *Principe of Datamining.* MIT Press, 2001.

[17] D.Michie. « Methodologies from Machine Learning in Data Analysis and Software ». In : *The computer journal* 34 (1991), p. 559–565.

[18] F.Eibe et K.Morgan. *Datamining.* 3rd-Edition, 2012, p. 664.

[19] F.Valdez, P.Melin et O.Castillo. « A New Evolutionary Method Combining Particle Swarm Optimization and Genetic Algorithms Using Fuzzy Logic ». In : *Soft Computing for Hybrid Intelligent Systems* (), p. 347–361.

[20] Gandelli et al. « Genetical swarm optimization : an evolutionary algorithm for antenna design ». In : *Journal of Automatika* 47 (2006), p. 3–4, 105–112.

[21] G.H.John. « Enhancements to the Datamining process ». Doctorat. tanford USA, 1997.

[22] G.H.John, R.KOHAVI et K.PFLEGER. « Irrelerant features and the subset selection problem ». In : *Machine Learning : proceeding of the eleventh international* (1994), p. 121–129.

[23] G.John, R.Kohavi et K.Peger. « Irrelevant features and the subset selection problème ». In : *Eleventh International Conference on machine Learning* (1994), p. 121–129.

[24] H.Almuallim et T.G.Dietterich. « Learning with many irrelevant features ». In : *Proceeding of The Ninth National Conference on Artificial Intelligence(AAAI)* (1991), p. 547–552.

[25] H.Hachimi. « Hybridations d'algorithmes métaheuristiques en optimisation globale et leurs applications ». Doctorat. Ecole Mohamedia d'ingénieurs (Rebat,Maroc), 2013.

[26] H.Liu et H.Motoda. *Computational methods of feature selection*. USA : Taylor et Francis group, 2008.

[27] H.Liu et L.Yu. « Toward integrating feature selection algorithms for classification and clustering ». In : *IEEE Transaction On Knowledge* 17 (2005), p. 491–502.

[28] HS.Lopes et LS.Coelho. « Particle swarm optimization with fast local search for the blind travelling salesman problem ». In : *Proceedings of fifth international conference on hybrid intelligent systems* (2005), p. 6–9.

[29] I.H.Osman et G.Laporte. « Metaheuristics : A bibliography ». In : *altzer Science Publishers* 63 (1996), p. 513–623.

[30] I.Inza et al. « Filter versus wrapper gene selection approaches in DNA microarray domains ». In : *Artificial Intelligence in Medicine* 31 (2004), p. 91–103.

[31] J.Achtnig. « Particle Swarm Optimization with Mutation for High Dimensional Problems ». In : *Engineering Evolutionary Intelligent Systems-Springer* (2008), p. 423–439.

[32] J.Czogalla et A.Fink. « On the Effectiveness of Particle Swarm Optimization and Variable Neighborhood Descent for the Continuous Flow-Shop Scheduling Problem ». In : *Metaheuristics for Scheduling in Industrial and Manufacturing Applications* (2008), p. 61–89.

[33] J.E.Hunt et D.E. Cook. « Learning using an artificial Immune Systems ». In : *Journal of network and computer applications* 19 (1996), p. 189–212.

[34] J.Habibi, SA.Zonouz et M.Saneei. « A hybrid PS-based optimization algorithm for solving traveling salesman problem ». In : *IEEE symposium on frontiers in networking with applications* (2006), p. 18–20.

[35] J.Han et M.Kamber. *Datamining : Concepts and techniques*. 3rd-Edition. USA, 2000, p. 703.

[36] J.H.Holland. « Genetic Aalgorithms and the optimial allocation of trials ». In : *SIAM Journal of Computing* 2 (1973), p. 88–105.

[37] J.Kennedy et R.C.Eberhart. « A discrete binary version of the particle swarm algorithm ». In : *Proceedings of the World Multiconference on Systemics, Cybernetics and Informatics, NJ* (1997), p. 4104–4109.

[38] J.P.Benzecri. *L'analyse des données*. 3e édition. Paris : Dunod, 1979.

[39] J.Quinlan et M.Kaufmann. *Programs for Machine Learning*. USA, 2014, p. 302.

[40] J.Robinson, S.Sinton et Y.R.Samii. « Particle swarm, genetic algorithm, and their hybrids : optimization of a profiled corrugated horn antenna ». In : *Proceedings of the IEEE International Symposium in Antennas and Propagation Society* (2002), p. 314–317.

[41] J.Sun et al. « Quantum-Behaved Particle Swarm Optimization with Binary Encoding ». In : *Center of Intelligent and High Performance Computing* (2007), p. 376–385.

[42] J. Kennedy et R.C. Eberhart. « Particle Swarm Optimisation ». In : *Conference Proceeding of neural networks*. Sous la dir. de Perth. Australia : IEEE, nov. 1995, p. 1942–1948.

[43] K.Premalatha et A.M.Natarajan. « PSO with GA Operators for Document Clustering ». In : *International Journal of Recent Trends in Engineering* 16 (2009).

[44] Daniel T. Larose. *Discovering Knowledge in data*. Canada, 2005.

[45] L.C.Molina, L. Belanche et A. Nebot. « Evaluating feature selection algorithms ». In : *CCIA-LNCS* (2002), p. 216–227.

[46] L.Lebart, M.Piron et A.Morinreau. *Statistique exploration miltidimensionnelle :Visualisation et inférience en fouille de données*. 4ème edition. Paris : Dunod, 2006, p. 480.

[47] L.N.Castro et J.Timmis. *Artificial immune Systems : A New computational Intelligence Approach*. London : Springer-Verglas, 2002.

[48] M.Dash et H.Liu. « Feature selection for classification ». In : *Intelligent Data analysis 1* (1997), p. 131–156.

[49] M.Dorigo et T.Stützle. *Ant Colony Optimization*. Cambridge : MIT Press, 2004.

[50] M.Goyal et R.Vohra. « Applications of Data mining ». In : *Higher Education* 9 (2012), p. 13–37.

[51] M.Jambu. *Introduction au Datamining*. Eyrolles, 1998, p. 136.

[52] M.S.Kiran et al. « A novel hybrid approach based on Particle Swarm Optimization and Ant Colony Algorithm to forecast energy demand of Turkey ». In : *Energy Conversion and Management* 53 (2012), p. 75–83.

[53] N.Benahmed. « Optimisation de réseaux de neurones pour la reconnaissance de chiffres manuscrits isolés : Sélection et pondération des primitives par algorithmes génétiques ». Mém.de mast. Université du Québec Canada, 2002.

[54] N.Cristianini et John J.Shawe-Taylor. *An Introduction to Support Vector Machines : And Other Kernel-based Learning Methods*. New York, NY, USA : Cambridge University Press, 2000.

[55] N.Metropolis et al. « Equation of state calculations by fast computing machinesg ». In : *Journal of Chem Physics* 21 (1953), p. 1087–1092.

[56] N.Mladenovic et P.Hansen. « Variable Neighborhood Search ». In : *Computers an Operations Research* 24 (1997), p. 1097–1100.

[57] N.Vladimir et Vapnik. *The nature of statistical learning theory*. New York : Springer-Verlag, 1995.

[58] PJ.Angeline. « Using selection to improve particle swarm optimization ». In : *Proceedings of the IEEE congress on evolutionary computation* 34 (1998).

[59] P.Somol, J.Novovicova et P.Pudil. « Advances In Feature Selection Methodology : An Overview of Recent Utia Results ». In : *Kybernetika* 4 (2004).

[60] P.S.Shelokar et V.K.Jayaramen. « Particle swarm and ant colony algorithms hybridized for improved continuous optimization ». In : *Applied Mathematics and Computation* 188 (2007), p. 129–142.

[61] R.Agrawala et A.Srikant. « Fast algorithms for mining association rules ». In : *IBM Almaden Research Center* (1994), p. 487–499.

[62] R.C.Eberhart, P.Simpson et R.Dobbins. « Computational PC Tools ». In : *AP Professional* (1996), p. 212–226.

[63] R.Féraud et al. « The orange customer analysis platform ». In : *Industrial Conference on Data Mining,Springer* (1997), p. 584–594.

[64] R.H.Creecy et al. « Trading mips and memory for knowledge engineering ». In : *ACM* 35 (1992), p. 48–64.

[65] R.Lefebure et G.Venturi. *Le Datamining*. Eyrolles, 2001.

[66] S.Das. « wrappers and a boosting-based hybrid for feature selection ». In : *Proceedings of the Eighteenth International Conference on Machine Learning* (2001).

[67] S.Das, A.Abraham et A.Konar. « Metaheuristic Pattern Clustering-An Overview ». In : *Metaheuristic Clustering, Studies in Computational Intelligence* 178 (2009), p. 257–264.

[68] S.Davies et S.Russell. « NP-completeness of searches for smallest possible feature sets ». In : *AAAI Fall Symposium on Relevance* (1994), p. 37–39.

[69] U.M.Fayyad et al. « Advances in knowledge discovery and Datamining ». In : *AAAI Press MIT Press* 17 (1996), p. 317–333.

[70] X.Shen et al. « A dynamic adaptive dissipative particle swarm optimization with mutation operation ». In : *Proceedings of IEEE/ICCA* (2007), p. 586–589.

[71] Y.Bennani. « Système d'apprentissage connexionnistes : sélection de variables ». In : *Intelligence Artificielle* 15 (2001), p. 3–4.

[72] Y.Grandvalet et S.Canu. « Adaptive Scaling for feature selection in svms ». In : *Proceeding of Advances in Neural Information Processing Systems (NIPS)* (2002), p. 553–560.

[73] YM.Clerc et J.Kennedy. « The particle swarm : explosion, stability, and convergence in multidimensional complex space ». In : *IEEE Transactions on Evolutionary Computation* 6 (2002), p. 58–73.

[74] Y.N.Zhang, Q.N.Hu et H.F.Teng. « Active target particle swarm optimization ». In : *Journal of concurrency computation practices and experiences* 20 (2005), p. 29–40.

[75] Y.Shi et R.C.Eberhart. « Parameter selection in particle swarm optimization ». In : *Evolutionary Programming VII* (1998), p. 591–600.

[76] Z.Lian, X.Gu et B.Jiao. « A similar particle swarm optimization algorithm for permutation flow shop scheduling to minimize makespan ». In : *Applied Mathematics and Computation* (2006), p. 773–785.